NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

A MULTI-ARMED BANDIT APPROACH TO FOLLOWING A MARKOV CHAIN

by

Ezra W. Akin

June 2017

Thesis Advisor:	Roberto Szechtman
Second Reader:	Moshe Kress

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704–0188

1. AGENCY USE ONLY *(Leave Blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	June 2017	Master's Thesis 06-30-2015 to 06-16-2017

4. TITLE AND SUBTITLE

A MULTI-ARMED BANDIT APPROACH TO FOLLOWING A MARKOV CHAIN

5. FUNDING NUMBERS

6. AUTHOR(S)

Ezra W. Akin

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Postgraduate School
Monterey, CA 93943

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

N/A

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

The views expressed in this document are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol Number: N/A.

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release. Distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

Across defense, homeland security, and law enforcement communities, leaders face the tension between making quick but also well informed decisions regarding time-dependent entities of interest. For example, consider a law enforcement organization (searcher) with a sizable list of potential terrorists (targets) but far fewer observational assets (sensors). The searcher's goal being to follow the target, but resource constraints make continuous coverage impossible, resulting in intermittent observational attempts. We model target behaviour as a discrete time Markov chain with the state space being the target's set of possible locations, activities, or attributes. In this setting, we define "following the target" as the searcher, at any given time step, correctly identifying and then allocating the sensor to the state which has the highest probability of containing the target. In other words, in each time period the searcher's objective is to decide where to send the sensor, attempting to observe the target in that time period, resulting in a hit or miss from which the searcher learns the target's true transition behaviour. We develop a Multi-Armed Bandit approach for efficiently following the target, where each state takes the place of an arm. Our search policy is five to ten times better than existing approaches.

14. SUBJECT TERMS

applied probability, stochastic optimization, machine learning, discrete time Markov chains, stochastic Multi-Armed Bandit, combinatorial Multi-Armed Bandit, online learning, and intelligence analysis.

15. NUMBER OF PAGES 75

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2–89)
Prescribed by ANSI Std. 239–18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

A MULTI-ARMED BANDIT APPROACH TO FOLLOWING A MARKOV CHAIN

Ezra W. Akin
Captain, United States Marine Corps
B.S., United States Naval Academy, 2009

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
June 2017

Approved by: Roberto Szechtman
Thesis Advisor

Moshe Kress
Second Reader

Patricia Jacobs
Chair, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Across defense, homeland security, and law enforcement communities, leaders face the tension between making quick but also well informed decisions regarding time-dependent entities of interest. For example, consider a law enforcement organization (searcher) with a sizable list of potential terrorists (targets) but far fewer observational assets (sensors). The searcher's goal being to follow the target, but resource constraints make continuous coverage impossible, resulting in intermittent observational attempts. We model target behaviour as a discrete time Markov chain with the state space being the target's set of possible locations, activities, or attributes. In this setting, we define "following the target" as the searcher, at any given time step, correctly identifying and then allocating the sensor to the state which has the highest probability of containing the target. In other words, in each time period the searcher's objective is to decide where to send the sensor, attempting to observe the target in that time period, resulting in a hit or miss from which the searcher learns the target's true transition behaviour. We develop a Multi-Armed Bandit approach for efficiently following the target, where each state takes the place of an arm. Our search policy is five to ten times better than existing approaches.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

KKT	Karush-Kuhn-Tucker
KTO	Known Transition Dynamics with Oracle
MAB	Multi-Armed Bandit
MABA	Multi-Armed Bandit (MAB) Allocation
NPS	Naval Postgraduate School
OCO	Online Convex Optimization
PDF	Probability Density Function MLE Maximum Likelihood Estimation
UCB	Upper Confidence Bound
TS	Thompson Sampling
UAV	Unmanned Arial Vehicle
UTO	Unknown Transition Dynamics with Oracle

THIS PAGE INTENTIONALLY LEFT BLANK

Executive Summary

Across the defense and homeland security communities, decision makers are faced with the tension between making quick but also well-informed decisions regarding issues of interest that change over time.

Consider, for example, a law enforcement organization with a sizable list of potential terrorists but a limited number of observational assets. We designate these potential terrorists as targets and the observational assets as sensors. The sensors being patrol officers, cameras, or even small drones. Because of the disparity between the number of targets and available sensors, continuous coverage is impossible resulting in intermittent observational attempts (hourly, daily, or even weekly). The goal of the law enforcement organization, the searcher, is to learn the baseline behaviour pattern for each target as quickly as possible. Once a reasonable baseline is established, the searcher would shift to some form of change-point detection, in order to detect if the target is planning an attack or just going on vacation.

In this thesis we examine how to quickly establish a behaviour pattern baseline, providing a method that consistently outperforms the Naïve version in expectation. We model the target's behavior as a discrete time Markov chain. The state space being the target's location, activity, or any specific attributes that change with time. In the simplest scenario, the searcher has one sensor and in each time period decides where to send the sensor, attempting to find the target, and resulting in a hit or miss from which the searcher learns the target's transition behaviour. In general, the searcher's decision variables are the sensor's locations (i.e., states of the Markov chain) over time. The searcher's objective is to allocate the sensor dynamically so as to learn the target's behaviour pattern as quickly as possible. Figure 1 depicts a simple four state example transition kernel that defines the behaviour pattern for a target. We focus on the House to Café transition ($p_{1,4}$).

We develop a Multi-Armed Bandit approach for efficiently following this target, where each state takes the place of an arm. Our search policy is five to ten times better than existing approaches as can be seen in Figure 2. This figure corresponds to our method's performance on the target defined in Figure 1. The black line being the true probability and the blue line being our method's estimate at each time step [n].

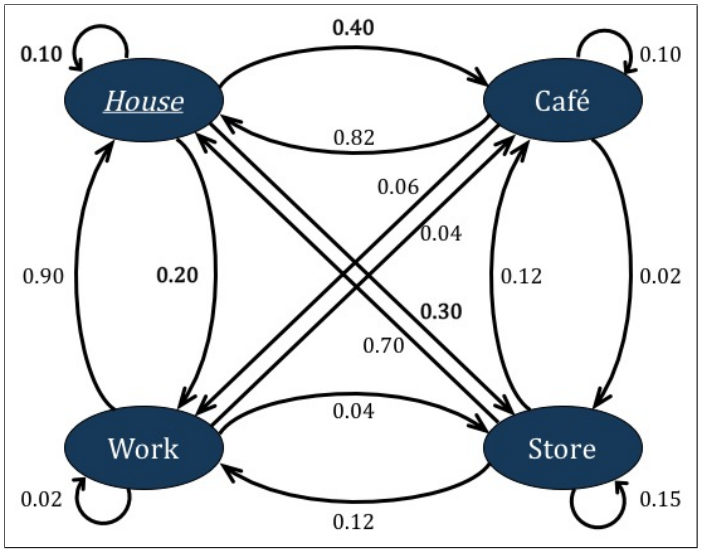

Figure 1. Example Four State Transition Probability Graph. The House to Café Transition Being Bolded and Corresponding to the State 1 to State 4 Transition.

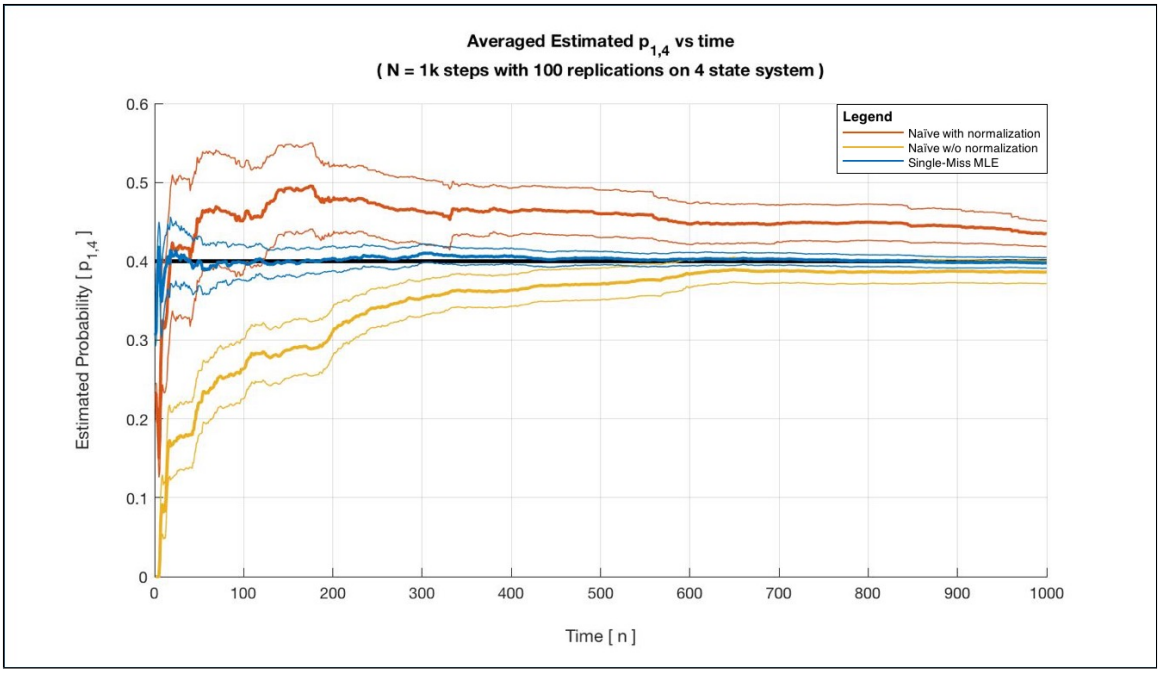

Figure 2. Estimated Four State Transition Probabilities (Focused on the House to Café or State 1 to 4 Transition) vs. Time (Mean with 95% Confidence Intervals). Comparison Between Current Naïve Methods and our Single-Miss MLE Approach. Generated in MATLAB.

Acknowledgments

I would first like to acknowledge my dependence on the Lord and thank Him for His constant love and guidance.

I would also like to express the sincerest gratitude to my parents, especially my mother, for inculcating into me a love of learning and sense of duty to country.

A special thanks to my best friend and wife, Abby, for her boundless patience, constant love, and heartfelt support during this journey, and to my wonderful children who reminded me that a world exists outside the bounds of homework and thesis writing!

Finally, I must express the deepest appreciation and indebtedness to the faculty of the Operations Research Department, particularly my advisor, Roberto Szechtman, and second reader, Moshe Kress. Their deep knowledge, patience, humility, assistance, and guidance were crucial in this endeavor, and I will value their friendship the rest of my life.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

This chapter provides the backdrop for this thesis. It not only defines the problem, why it is important, and how it is currently being solved, but it also provides the ground rules.

1.1 The Problem

We consider a situation where a searcher attempts to locate and maintain observation of a target (e.g., terrorist, pirate ship, aircraft debris, endangered animal, or generically, a target of interest). We model the target's behavior as a discrete time Markov Chain (a layman's explanation of this can be found in Chapter 2, Section 2.1). The state space being the target's location, activity, or any specific attributes that change either with time or in some discrete or sequential fashion (e.g., bank account activity). In particular, the states can be physical locations, radio communication activity, the IP address of the computer used by the target, or even the target's current bank account levels or some specific flagged expenditures.

In the simplest scenario, the searcher has one sensor to follow the entity of interest. For instance, at time $t = 7$ the target can be in locations (or generically, states) a, b, or c. If the searcher, at $t = 7$, allocates the sensor to location c and the target also transitioned to that state, then the searcher earns a reward of one. The searcher's decision variables are the sensor's locations (i.e., states of the Markov chain) each time step, using past sensor response information to guide future decisions.

The objective of the searcher (the entity attempting to follow the target) is to allocate the sensor dynamically so as to achieve as close to constant observation (sensor positively observing the target at each time step) as possible. If the target's transition matrix was known, then the searcher would simply position the sensor on the state with the highest probability given the last observed state of the target (which is not necessarily the last time period) and the sequence of observed states which failed to reveal the target (the states that did not contain the target or the sequence of misses). In this thesis, we relax the assumption of a known transition matrix. Hence, the searcher attempts to learn the underlying Markov transition matrix of the target, constant observation being the goal.

1.2 Motivation: Why It Is Important

As briefly mentioned in Section 1.1, our problem can be applied to a very broad range of topics or settings where some form of transition dynamics need to be learned. Here it is sufficient to highlight a couple that will serve as surrogates for the rest of the possible application areas. Specifically, we will expand upon our primary setting of a terrorist in a city as well as possible applications to learning the migratory patterns of endangered animals (birds being our choice of example) and identifying potential pirates hiding within a fleet of innocent fishing vessels.

Suppose there is a person of interest (the target) living in a large city and the authorities need to determine his/her behavior patterns in order to efficiently (minimum number of assets and as quickly as possible) track or maintain observation. However, in a typical scenario there exist many such targets, meaning that resource limitations preclude persistent surveillance. To efficiently track the target, the searcher employs various sensors depending on where it is believed the target is currently located. Another reason for intermittent search is to prevent the target from realizing he is being tracked. Authorities might therefore intentionally limit the use of sensors following the target, while attempting to periodically regain observation of the target. This observation might be having a human asset walk by the front of a cafe, glancing inside to see if the target is there or not. We model the target's behavior as a discrete time Markov Chain with the state space comprising of all the various physical locations that the target might visit. These might be static locations such as the cafe mentioned above or the terrorist's house but could also include more dynamic states such as the target being in a vehicle or on the subway. This setting will serve as our primary motivation, and thus it is important to keep it in mind to provide a framework on which our algorithms will be built.

1.2.1 Migratory Patterns of Endangered Birds

Instead of attempting to follow a potential terrorist in a city (the person of interest or target), imagine the goal is to learn the migratory patterns of an endangered bird. In this setting we would model the state space primarily to cover such things as latitude and longitude (each state corresponding to a physical location and a small set of activities). The searcher would then apply our approach on this specific flock of birds, learning over time the basic seasonal patterns of this bird species.

1.2.2 A Pirate Hiding in a Fishing Fleet

In this setting, which is much closer to our primary scenario, the searcher is attempting to learn the overall behavior pattern of a fishing fleet in order to determine if a pirate is hiding in the fleet. In this setting the searcher would replicate our algorithm for each "fishing" vessel, updating these algorithms with a drone or Unmanned Arial Vehicle (UAV) network or mesh. Each UAV would update all algorithms based on which vessels it can see during that time step.

1.3 Current Solution Methods

An important question that needs to be raised though is, how is this problem currently being solved? Essentially, if this thesis did not exist, what would people use instead? While an answer to this question, by nature, cannot be comprehensive it will still be beneficial to examine how someone might attempt to learn the behaviour pattern of a terrorist within a city without the algorithms developed in this thesis. The naïve (not intended here as derogatory) approach to following a target is to estimate the transition matrix by observing the target's transitions. For example, if the searcher knows the target is currently in state a, the searcher would then look in state b during the next time step. Every time the searcher observes or fails to observe the target, more information is gained. After accumulating a number of hits and misses (assuming stationary target behaviour), the searcher can use the current time step's estimated transition matrix to decide where to put the sensor next. The naïve method for generating this estimated transition matrix is to use the ratio of hits (observed transitions) to total attempted transition observations as the probability for the target to make that specific transition.

Initially of course, the searcher will not have a sequence of hits and misses for use in estimating the target's behaviour so will have to begin with pure exploration (i.e., assume uniform probabilities as the first estimate). Later on, after obtaining a sequence of hits and misses, placing the sensor in the most likely target's location (pure exploitation) may result in poor performance. A more sophisticated approach would include a term to force exploration of the whole state space (so that learning takes place). This and other approaches are fleshed out in Chapter 3: Methodology. We develop an algorithm that includes the sequence of hits and misses plus a judiciously chosen inflation term to force exploration that consistently learns faster.

1.4 Our Approach

Our goal is to develop an algorithm that learns faster than the naïve approach. Specifically, we apply methods from the following fields of research: Applied Probability, Optimization, Online Learning, and Statistics. We start with an ideal situation where we not only know the transition dynamics (or probabilities) but also have some form of Oracle which provides us the actual location or state the target transitioned to if we miss it. We then sequentially relax the assumptions for this situation until we get to the point where we do not know the transition dynamics at all and do not have an Oracle. This of course is more fully discussed in Chapter 3: Methodology.

Our algorithm leverages not only the ratio of hits over total observation attempts (hits and misses) for a single transition probability but also the fact that each miss contains quite a lot of information regarding other transitions. This additional information comes from the assumption that we designed the state space to be comprehensive. In other words, if the target is currently at state a, it must either stay in that state or transition to another state within the state space. We assume that the target cannot jump out of the state space (i.e., we have defined the state space to be exhaustive).

Therefore, in the situation just described, if the searcher allocates the sensor to look in state a again but the target is not there, we can distribute the weight of this miss (in the naïve approach this means just increasing the denominator of the ratio; in our approach this weighting is calculated by an optimization problem) across all the rest of the states (as a miss for the a to a transition but as a partial hit for the a to x transition, x being all non-a states).

Intuitively, the two-state example is the easiest to see. If our state space is $\{a, b\}$ and the target is currently in state a, then if it does not transition back to a we know implicitly that it must have transitioned to b. Therefore, instead of just updating the a to a transition probability with a miss, we also update the a to b transition probability with a hit. In the two-state setting a miss provides just as much information as a hit.

1.5 Measures of Success

How will we know if we have succeeded? We measure our success by our algorithm's expected regret as compared to the naïve approach mentioned above in Section 1.3. We define expected regret as the cumulative difference between our estimate of the target's transition probability and the true probability over time. Further, we seek to provide upper bounds (i.e., worst case bounds) on the performance of our algorithm, namely on the expected regret growth over time. This is explained in more detail in Chapters 4 and 5.

1.6 Ground Rules: Scope, Limitations, and Assumptions

Before going any further, it will be helpful to lay out a scope for this thesis, some limitations, and a few assumptions we are making in our approach. In this thesis we develop an algorithm that utilizes and optimizes over the one-step misses (hence, the name "Single-Miss MLE") returned from a sensor which only provides binary responses, as mentioned in Section 1.2. We define a one-step miss as the situation where the searcher knows the location of the target in the previous time step but missed it in the current time step. If the searcher missed it again in the next time step, that would be a two-step miss.

Additionally, we assume that the state space is comprehensive (i.e., the target cannot "jump" out of the state space); we only examine sensors that have neither false positive nor false negative rates; we only consider discrete time; we assume that the target's behavior is stationary (i.e., the transition matrix doesn't change over time); we only consider one target and one sensor; and we assume that the sensor is unobserved by the target. Relaxing any of these last assumptions or limitations would make very good extensions and we hope to explore them in future work.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2:
Background and Literature Review

The purpose of this chapter is to orient the reader to where this thesis research fits into the broader discipline of Operations Research as well as provide a common background of concepts that we will leverage and build upon. The topics or areas of research that intersect with our problem are: Persistent Surveillance, Search and Detection Theory, Markov Chains from Probability Theory and, Online Learning, specifically the Multi-Armed Bandit (MAB) approach to Machine Learning. The first two are parallel efforts that we wish to assist by attacking our problem from an Online Learning approach using the setting of a Markov Chain that will provide a more general range of settings than usually seen.

2.1 Markov Chains

In this section we refresh the reader on some of the basics regarding Markov Chain theory. This is necessary as we intend to leverage the power and flexibility of the Markovian approach enabling us to effectively model a wide range of real-world search and observation problems. Much of this material is a summary from the incredibly useful "Probability Models for Practitioners" [1] class notes written by Professor Kyle Lin from the Naval Postgraduate School (NPS).

Here we take a moment to define a Markov Chain for the reader. A discrete time Markov Chain is a sequence of random variables $X_1, X_2, \ldots$, indexed by time taking values in some state space, with the property that future values $X_{t+1}, X_{t+2}, \ldots$ are only dependent on the current state $X_t = x$, and therefore conditionally independent of the past. We call this independence the **Markov property**. While this may seem like a rather large assumption to make, if needed, we can embed information about the past into the current state thereby maintaining this assumption without losing the mathematical power of the memoryless property of Markov Chains.

More formally, a discrete time Markov chain is a stochastic process $(X_t : t = 1, 2, \dots)$ taking values in a discrete **state space** $S = \{1, 2, \dots, s\}$, that satisfies the **Markov property**, meaning that

$$P(X_{t+1} \in A \mid X_1, \dots, X_t) = P(X_{t+1} \in A \mid X_t),$$

for $A \subseteq S$, so that the distribution of X_t depends on the past only through X_{t-1}. Appendix B provides a simple example of a discrete time Markov Chain.

2.2 Machine Learning

In this section, we delve into the topic of Machine Learning, specifically its sub-discipline, Online Learning. We give an overview of Online Learning and then delve into some specific methods that we use in this thesis including the MAB Problem, Thompson Sampling (TS), the classical Maximum Likelihood Estimation (MLE) Point Estimation method, and finally the Upper Confidence Bound (UCB) algorithm for the Stochastic MAB problem. We include a short discussion of MLE because it is a critical component in our approach to estimating a given probability based on a sequence of data.

2.2.1 Online Learning Overview

Online Learning or Online Convex Optimization (OCO) is a sub-discipline of Machine Learning under which it was first defined. Hence, it primarily studies the performance of learning algorithms. As indicated by the second title, at heart it is optimization within a dynamic setting vice the standard deterministic setting and therefore has very broad applicability. As succinctly stated by Hazan, in his *Introduction to Online Convex Optimization* [2]:

> In many practical applications the environment is so complex that it is infeasible to lay out a comprehensive theoretical model and use classical algorithmic theory and mathematical optimization. It is necessary as well as beneficial to take a robust approach: apply an optimization method that learns as one goes along, learning from experience as more aspects of the problem are observed.

Of note, this conceptualization blurs the classic definitions or boundaries of deterministic modeling, stochastic modeling, and optimization methodologies [2]. As can be seen from the dynamic setting, the algorithm doesn't have all of the information in the beginning but still must act.

The basic framework, from Hazan [2], is that a player (the searcher in our setting) makes iterative decisions while online (think making decisions with partial information and additional information arriving as a stream). The underlying game structure is explained later in this section. When the player makes each decision, the outcomes (think penalty or loss) associated with those possible choices are unknown. Once the player commits to a choice, he/she will receive the amount of loss associated with that specific choice. Unlike Dynamic Programming, the player does not know in advance the losses associated with a given decision. Again, as mentioned above, these losses are unknown to the player prior to making his/her decision. While these losses can be dependent on the player's choices, they could also be assigned by an adversary or opponent! The following restrictions must therefore be imposed to make this framework feasible and complete:

1. The **losses** associated with the set of choices must be bounded. Otherwise, the adversary could decrease the scale of losses each iteration such that the player (algorithm) would never recover from the first loss. Therefore, the losses must reside within a bounded region.

2. The **set of decisions** facing the player or algorithm must be somehow bounded or structured. This ensures that we have some sort of meaningful performance metric and prevents the adversary from assigning large losses to each choice made by the player (algorithm) indefinitely while separating a set of strategies that have no loss.

Essentially, this framework can be viewed as a structured and repeated game [2]. The following notation helps solidify this framework. The set of decisions is a convex, nnonempty, bounded, and closed set in Euclidean space $\mathcal{K} \subseteq \mathbb{R}^n$ with the costs being modeled as bounded convex functions over $\mathcal{K}$ [2]. At iteration $t \in T$, T being the total number of game iterations, the player faces the set of decisions $x_t \in \mathcal{K}$. After committing to a choice, the associated cost function is displayed or revealed: $f_t \in \mathcal{F} : \mathcal{K} \to \mathbb{R}$. Hazan further defines $\mathcal{F}$ as being the bounded family of cost functions available to the adversary. The cost that

the player must now pay is $f_t(x_t)$. Our performance metric, taken from game theory, is the sum of the regret or difference between the lowest possible cost (from the cost function) and the one actually incurred by the player each iteration. We define this as the **regret** [2]. To formally define regret, consider an OCO algorithm, $\mathcal{A}$ that maps an online player's game history to a specific decision in the set of decisions over time [2]. This player's or algorithm $\mathcal{A}$'s regret after T iterations is formally defined by Hazan in [2], Equation 1.1, as:

$$\text{regret}_T(\mathcal{A}) = \mathcal{R}_T(\mathcal{A}) = \sup_{\{f_1,\ldots,f_t\}\subseteq\mathcal{F}} \left\{ \sum_{t=1}^{T} f_t(x_t) - \min_{x\in\mathcal{K}} \sum_{t=1}^{T} f_t(x) \right\} \tag{2.1}$$

From this definition of regret, we see that it is desirable for the regret to be sub-linear as a function of time or T. This setting or framework for online learning has become very popular recently primarily due to its powerful modeling capabilities [2]. Specifically, it can be used to model such diverse problems as online routing, advert placement and selection, and even spam filtering [2].

2.2.2 The Multi-Armed Bandit (MAB) Problem

In this sub-section we delve further into the Online Learning framework with a specific, and rather popular variant, the Stochastic MAB. The Stochastic MAB is named after the "Single-Arm Bandit," a Vegas slot machine, which still serves as one of the best ways to describe this problem. Of note up front, much of the material from this section is a summary of or taken directly from Mahajan and Teneketzis' *Multi-armed Bandit Problems* [3] and Agrawal and Goyal's *Analysis of Thompson Sampling for the Multi-Armed Bandit Problem* [4].

Imagine a player entering a casino and purchasing 20 "tokens" with which to play a row of "Single-Arm Bandits." We call this row of slot machines a "Multi-Armed Bandit" with each slot machine's lever or handle being an "Arm" of this MAB. Further, we assume that each slot machine or arm has the potential to have a payout or reward (based upon a potentially different underlying probability distribution for each arm) and each arm's reward is a realization or sample from that distribution. Since the player, does not initially know which arm will give the best payout (i.e. has the most lucrative reward distribution), the player must begin by exploring the system, using a few tokens to compare the arms. At this point, the player has very rough estimates on the potential rewards from a few arms and

must decide to either *exploit* the best arm so far or continue to *explore* the rest of the arms with the finite tokens. This tension between *exploration* and *exploitation* is the heart of the MAB Family of Problems.

Another way of thinking about this tension or describing this problem is to imagine a player with a single resource at each finite time step with which to allocate to one of a number of competing projects. Upon allocating the resource, that project changes while the rest stay static. Further, the reward or return on investment from that project is different than what each of the other projects might have returned. Hence, the MAB problems are a class of sequential resource allocation problems [3]. Further, most MAB algorithms use the player's regret as their measure of effectiveness. This regret is essentially the same as that defined above in the Online Learning section. It is defined as the difference between the "best" arm that the player could have pulled that time step, had he known all of the distributions, and the one he/she actually pulled. Hence, the regret.

There are of course many variants of this problem or ways to adjust the classic setup to fit numerous real-world situations such as: one or multiple resources available for allocation, new projects appearing over time, all projects changing each time step, or even an adversary who chooses the rewards of each arm.

We base our formulation of the MAB problem on Agrawal and Goyal [4]. Consider a casino with $\mathcal{K}$ slot machines, each of these "arms" denoted by $i \in \mathcal{K}$. At each discrete time step, $n = 1, 2, 3, \ldots$, the player must decide which of the $\mathcal{K}$ arms to pull. Each arm, i, returning a random, positive, real-valued, reward with support on [0, 1]. The rewards returned from each arm, immediately after pulling that arm, are independent and identically distributed as well as independent of the play of the other arms. Therefore, the player or MAB algorithm must decide which arm to pull, at time n, based on the rewards received (or in other words, the information obtained) up through time $n - 1$. Next, we define μ_i as the (unknown) expected or average reward for arm i, $i(n)$ as the specific arm played at time step n, and $\mu_{i(n)}$ as the (unknown) expected or average reward for the arm i pulled at n.

The goal therefore, is to maximize the total expected reward by time N. We denote this expected total reward by: $\mathbb{E}\left[\sum_{n=1}^{N} \mu_{i(n)}\right]$. Since this measure doesn't really tell us how well our algorithm performs over time (due to no comparison with how well we could have done)

we instead use the equivalent measure of expected total *regret*. This regret, as mentioned above, is the difference between the optimal arm and $i(n)$, the arm we played. To define this regret, $\mathcal{R}$, let $\mu^* := \max_i \ \mu_i$, $\Delta_i := \mu^* - \mu_i$ (which is always greater than or equal to zero by definition), and further, let $k_i(n)$ be the number of times the algorithm has pulled arm i by time n. Finally, we formally define this regret as follows:

$$\mathbb{E}[\mathcal{R}(N)] = \mathbb{E}\Bigg[\sum_{n=1}^{N}\Big[\mu^* - \mu_{i(n)}\Big]\Bigg] = \sum_{i}\Delta_i \cdot \mathbb{E}[k_i(N)] \tag{2.2}$$

2.2.3 Thompson Sampling (TS)

We will now examine the basic TS algorithm for the Bernoulli Bandit problem. This section's material and notation (notation does not follow previous sections) is from Agrawal and Goyal's *Analysis of Thompson Sampling for the Multi-Armed Bandit Problem* [4].

First, we examine a special case of the above Stochastic MAB problem where we consider the situation where the return or result from pulling an arm is Bernoulli or binary, i.e. hit or miss, success or failure. We model this response as simply 0 or 1. So, for arm i, the probability of success or of getting the reward (reward = 1) is μ_i. This special case is called the Bernoulli Bandit algorithm. This algorithm uses Bayesian priors on the Bernoulli means, μ_i's, for which Agrawal and Goyal propose the Beta family of distributions. They propose this because the Betas have support on the interval (0,1), are continuous probability distributions, and also enable a very natural posterior update; in other words, the Beta and Bernoulli distributions form a conjugate prior structure. The following is a quick summary of Beta distributions, followed by an exploration of the proposed Bernoulli Bandit algorithm.

The Beta family of distributions, as mentioned above, are continuous probability distributions with support on the interval (0,1). Below is a plot of some of these distributions for various ranges of the parameters. Their Probability Density Function (PDF), Beta(α, β), with parameters $\alpha > 0$ and $\beta > 0$, is given by: $f(x; \alpha, \beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1}$, and their mean is given by $\mu_{\text{Beta}} = \frac{\alpha}{\alpha+\beta}$ [4]. As can be seen from this equation, the higher the α and β's are, the lower the variance. You can see this decreased variance in Figure 2.1, specifically, the gold distribution as compared to say the light blue one.

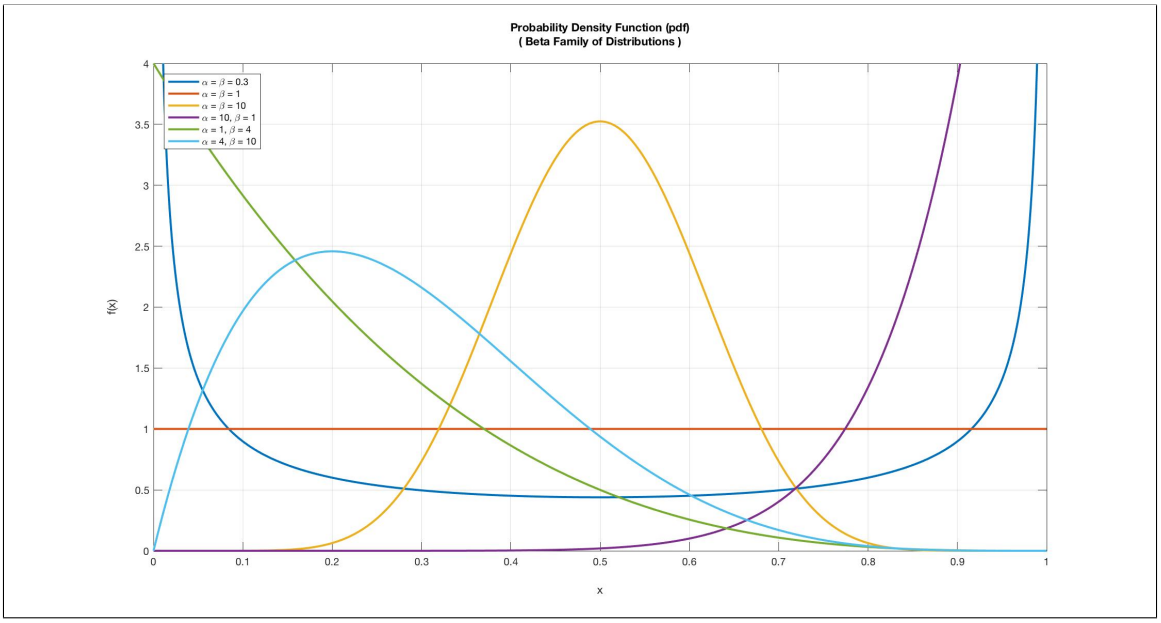

Figure 2.1. The Family of Beta Distributions

The basic TS algorithm uses the red distribution, in Figure 2.1, as the prior for each arm. Specifically, it assumes that arm i has a prior Beta(1,1) on μ_i. This is a natural choice as it essentially assumes a uniform distribution on the interval (0, 1) [4]. Of note, the following notation breaks from that defined in previous sections in order to more closely follow [4]. Next, at time t, having observed $S_{i(t)}$ successes or hits (think, reward of 1) and $F_{i(t)}$ failures or misses (think, reward of 0) in $k_{i(t)} = S_{i(t)} + F_{i(t)}$ plays of arm i, the algorithm updates the current distribution of μ_i to Beta($S_{i(t)} + 1, F_{i(t)} + 1$). Lastly, the algorithm samples from these posterior distributions for the means of the arms, μ_i's, and plays the arm with the highest probability of having a success or in other words, the largest mean. This method from Agrawal and Goyal [4] is summarized in Algorithm 1, found in Table 2.1.

Algorithm 1: Basic Thompson Sampling for Bernoulli bandits

For each arm $i = 1, 2, \ldots, N$ set $S_i = 0, F_i = 0$
for *$t = 1, 2, \ldots$* **do**
  for *each arm $i = 1, \ldots, N$* **do**
    Sample $\theta_i(t)$ from the Beta$(S_{i(t)} + 1, F_{i(t)} + 1)$ distribution.
  end
  Play arm $i(t) := \text{argmax}_i\ \theta_i(t)$ and observe reward r_t.
  if $r_t = 1$ **then**
    Set $S_{i(t)} = S_{i(t)} + 1$
  else
    Set $F_{i(t)} = F_{i(t)} + 1$
  end
end

Table 2.1. Algorithm: Basic Thompson Sampling

The basic idea behind this algorithm is that at each iteration or time step, the TS algorithm attempts to pull the arm with the largest probability of returning a reward of 1. The intuition here is that each reward of 1 increments the α parameter of the associated Beta distribution, shifting it closer to one, while each reward of zero increments the β parameter of the associated distribution, shifting it closer to zero. To see this graphically, look at the light blue and purple distributions in Figure 2.1. Additionally, as the number of samples increases the variance of the resultant Beta distribution decreases as mentioned in the last paragraph. This can also be seen in Figure 2.1, specifically, the gold distribution as compared to the light blue one. Further, because it constantly updates the estimated Beta distributions, the algorithm performs well (specific convergence bounds for this algorithm can be found in Agrawal and Goyal's paper, see [4]). This algorithm is then extended by Agrawal and Goyal to the general stochastic MAB setting (more information on this can also be found in their paper, see [4]).

2.2.4 Maximum Likelihood Estimation (MLE)

Here, we will take a moment to examine a very useful method for estimating a specific parameter from a distribution. In our case, we are trying to estimate an unknown transition probability from the underlying Markov Chain's transition probability matrix. The following summary and material is based on Jay Devore's textbook, *Probability and Statistics for Engineering and the Sciences* [5] as well as Erich Lehmann and George Casella's seminal text, the *Theory of Point Estimation* [6].

First introduced by R. A. Fisher, between 1912 and 1922, the method of MLE, as its name implies, attempts to accurately estimate a distribution's parameter(s) of interest given only a finite number of samples from that distribution. Specifically, the likelihood function provides us with how likely the observed samples are as a function of the possible parameter values. Then, by maximizing the likelihood function the MLE method returns the parameter values from which the observed data was most likely generated [5]. The following definition is taken directly from Devore [5]:

Definition: Maximum Likelihood Estimator

Let $X_1, X_2, \ldots, X_n$ have a joint probability mass function or PDF of

$$f(x_1, x_2, \ldots, x_n \mid \theta_1, \theta_2, \ldots, \theta_m) \tag{2.3}$$

where the parameters $\theta_1, \theta_2, \ldots, \theta_m$ have unknown values. When $x_1, x_2, \ldots, x_n$ are the observed sample values and (2.3) is regarded as a function of $\theta_1, \theta_2, \ldots, \theta_m$, it is called the **likelihood function**. The maximum likelihood estimates $\hat{\theta}_1, \hat{\theta}_2, \ldots, \hat{\theta}_m$ are those values of the θ_i's that maximize the likelihood function, so that

$$f(x_1, \ldots, x_n \mid \hat{\theta}_1, \ldots, \hat{\theta}_m) \;\geq\; f(x_1, \ldots, x_n \mid \theta_1, \ldots, \theta_m) \text{ for all } \theta_1, \ldots, \theta_m \tag{2.4}$$

When X_i's are substituted in place of x_i's, **maximum likelihood estimators** result.

Table 2.2. Definition: Maximum Likelihood Estimator. Reproduced from Devore's *Probability and Statistics for Engineering and the Sciences* [5]

But, why use the MLE method instead of some other option? As briefly mentioned by Devore in [5] and closely examined and proved by Lehmann in [6], the MLE method has some very useful and important properties making it a good choice. These properties are:

1. **Asymptotic Consistency**
   Under many conditions, MLEs converge in probability to the true parameter value, θ. Further, by increasing our sample size, n, we can also achieve an arbitrary level of precision [6].

2. **Asymptotic Efficiency**
   This means that as the sample size n increases (tends towards infinity) under certain conditions the MLE converges to the true parameter value, θ, as fast as the quickest possible method. In other words, this method converges as quickly as theoretically possible. It achieves the so-called Cramér-Rao lower bound, which means that no consistent estimator can converge more quickly [6], [7], [8]. While other consistent estimators may match an MLE in convergence rate, they are not able to beat it.

3. **Asymptotic Normality**
   Again, as n increases, the MLEs converge in distribution, under certain conditions, to a Gaussian (normal) distribution with the mean being equal to the true parameter value, θ, and a minimal variance [6]. Which, according to Devore is "as small as or nearly as small as can be achieved by any estimator." [5]

2.2.5 Upper Confidence Bound (UCB) Strategies

This section covers a critical strategy for developing theoretical upper-bounds on the MAB convergence rate or expected regret in a specific setting. This section's material and notation closely follows Bubeck and Cesa-Bianchi's *Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems* [9] as well as loosely following Elad Hazan's *Introduction to Online Convex Optimization* [2].

In order to examine this topic, it is necessary to start with some basic definitions from the Stochastic Bandit problem. From Bubeck and Cesa-Bianchi [9], each arm $i \in \{1, \ldots, K\}$ is tied to an unknown probability distribution v_i. The player, at each time step $t = 1, \ldots,$, picks an arm $I_t \in \{1, \ldots, K\}$ receiving a reward $X_{I_t,t}$ from the probability distribution v_{I_t}. This reward being independent of the past. Further, we denote the mean of arm i with μ_i and then define the optimal mean and arm below, which are unknown to the player: [9]

$$\mu^* = \max_{i=1,\ldots,K} \mu_i \qquad \text{and} \qquad i^* \in \operatorname*{argmax}_{i=1,\ldots,K} \mu_i$$

Next, we define pseudo-regret following Bubeck and Cesa-Bianchi [9] as:

$$\bar{R}_n = n\mu^* - \mathbb{E} \sum_{t=1}^{n} \mu_{I_t} \tag{2.5}$$

If i^* was known, then the agent would simply pull that arm in each iteration in order to minimize the pseudo-regret. Of note, the μ_{I_t}'s in Equation 2.5 are the actual means from those arms' distributions which are unknown by the agent.

We let $T_i(s) = \sum_{t=1}^{s} \mathbb{1}_{I_t=i}$, indicating the number of times that the player has selected arm i within the first s time steps, and we let $\Delta_i = \mu^* - \mu_i$, indicating the sub-optimality parameter of arm i (or the regret due to this arm having a larger penalty than the optimal arm). The idea is that pulling an arm with large Δ_i induces a large pseudo-regret. Of course, the agent doesn't know the values of Δ_i, but they exist and are well defined as long as the mean rewards are finite.

Following Bubeck and Cesa-Bianchi in [9], we assume that the distribution of rewards X are light-tailed and hence, there exists a convex function ψ on the real numbers such that, for all $\lambda \geq 0$,

$$\ln \mathbb{E}e^{\lambda(X - \mathbb{E}[X])} \leq \psi(\lambda) \qquad \text{and} \qquad \ln \mathbb{E}e^{\lambda(\mathbb{E}[X] - X)} \leq \psi(\lambda) \tag{2.6}$$

Following Bubeck and Cesa-Bianchi [9] further, the UCB algorithm can be applied to any light-tailed distribution, meeting the conditions defined in Equation 2.6, by forming an index for each arm, and then pulling the arm with the largest index estimated so far. Each index has two components. The first being the reward's sample mean obtained by pulling arm i for s times, $\hat{\mu}_{i,s} = \frac{1}{s}\sum_{t=1}^{s} X_{i,t}$. The second being an inflation term (this is the "upper confidence bound" part) selected so that the probability of a suboptimal index being larger than the optimal index is suitably small. In the following index, α is any positive constant, the optimal value calculation being examined in further detail in [9]. Also, from convex analysis we denote $\psi^*(\epsilon)$ as the Legendre-Fenchel transform of the function ψ as: $\psi^*(\epsilon) = \sup_{\lambda \in \mathbb{R}} (\lambda\epsilon - \psi(\lambda))$ where for example, if $\psi(x) = e^x$ then $\psi^*(x) = x\ln(x) - x$, $\forall x > 0$. Our index then, at time t, is defined as

$$\hat{\mu}_{i,T_i(t-1)} + (\psi^*)^{-1}\left(\frac{\alpha \ln t}{T_i(t-1)}\right)$$

for each arm i, where $\psi^*(\cdot)$ is the large deviations rate function corresponding to distribution v_i, and $T_i(t-1)$ is the number of pulls of arm i by time $t-1$. With this, one can show that

$$P\left(\hat{\mu}_{i,T_i(t-1)} - (\psi^*)^{-1}\left(\frac{\alpha \ln t}{T_i(t-1)}\right) > \mu_i\right) \leq t^{-\alpha},$$

which implies that the regret grows similar to $\log t$.

The question remains though, what is the function $\psi^*(\cdot)$? If, v_i has bounded support (the relevant scenario for this thesis) over $(-b, b)$, for $0 < b < \infty$, then, as Bubeck and Cesa-Bianchi mention in [9], one can use $\psi(\lambda) = \frac{b^2\lambda^2}{8}$. The Gaussian case – important for several applications – leads to $\psi(\lambda) = \frac{\lambda^2}{2\sigma^2}$, where σ^2 is the variance constant. It is possible to get suitable bounds for $\psi^*(\cdot)$ under further assumptions, but this lies outside the scope of this thesis.

In summary, the algorithm pulls arm

$$I_t \in \underset{i=1,\ldots,K}{\operatorname{argmax}} \left[\hat{\mu}_{i,T_i(t-1)} + (\psi^*)^{-1}\left(\frac{\alpha \ln t}{T_i(t-1)}\right)\right]$$

at time t. By using this strategy the algorithm achieves the following regret upper-bound as examined and defined by Bubeck and Cesa-Bianchi in [9]:

Theorem: Pseudo-regret of the (α, ψ)-UCB Strategy
Assume that the reward distributions satisfy Equation 2.6. Then (α, ψ)-UCB with $\alpha > 2$ satisfies $$\bar{R}_n \leq \sum_{i:\, \Delta_i > 0} \left(\frac{\alpha \Delta_i}{\psi^*(\frac{\Delta_i}{2})} \ln(n) + \frac{\alpha}{\alpha - 2} \right).$$

Table 2.3. Theorem: UCB Pseudo-Regret. Reproduced from Bubeck and Cesa-Bianchi's *Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems* [9]

The key intuition from this theorem, is that the sub-optimal arms are sampled at a natural logarithm of time rate while the optimal arm is sampled at a rate of time minus the natural logarithm of time. The inflation term therefore, ensures that every arm is sampled preventing the algorithm from getting "stuck" on a sub-optimal arm that just happened to have a long run of good rewards. It is well known that eliminating the inflation term from the index leads to a regret that grows linearly, as opposed to logarithmically.

2.3 Related Studies

The following are seven studies that utilize similar Machine Learning approaches to intelligence collection operations and are therefore worth mentioning even though their topics are focused on slightly different applications.

1. Costica in [10], examines methods for reducing the congestion generated during the most time consuming stages of the intelligence cycle, namely the classification stage. He proposes a tandem queue based optimization model as an analytic solution to this problem.

2. Nevo in [11], builds upon Costica's work by further studying this intelligence cycle bottleneck. He formulates the problem as an exploitation-exploration trade-off between good known intelligence sources and raw or unexplored sources that may or may not be valuable.

3. Ellis in [12], develops a software library implementing Nevo's previously generated mathematical model of information selection in an Online Learning setting, specifically, the intelligence cycle mentioned above. Further, he tests the performance of these different algorithms in a social communications network setting.

4. Tekin in [13], analyzes applying Online Learning methods to a couple of the early stages of the intelligence cycle, namely, the collecting and processing stages. He assumes that the intelligence products arrive sequentially such that Online Learning algorithms are a realistic approach. Specifically, he developes a modified Thompson Sampling algorithm to solve for the optimal arm to select given the most recent samples analyzed.

5. Marshall in [14], approaches the collection, processing, and analyzing stages of the intelligence cycle from a MAB Allocation (MABA) framework. Specifically, this framework models the problem as a "novel finite horizon Bayesian stochastic dynamic programming problem..." [14]. Further, he utilizes a novel Lagrangian based index heuristic for source, or arm, selection.

6. Hepworth in [15], investigates leveraging quantile, or superquantile, risk under a loss constraint in the context of a MAB setting. Specifically, he applies his algorithms in an intelligence collection setting where each arm of the MAB corresponds to a particluar item or document which may yield significant or little to no value to the intelligence analyst. He develops two sequential elimination algorithms which "select the most important source for a given constraint level, sampling from the arm(s) with the largest conditional expectation over a quantile" [15].

7. Grant in [16], attacks a significantly different problem then those listed above but utilizing similar methods. He focuses on the UAV Search Problem, specifically, allocating UAVs to various sub-regions or boundaries in order to optimize detection of events of interest. This problem is, of course, of great interest to the intelligence community at large. He approaches this problem along three broad avenues: Intensity Estimation, Optimization, and Machine Learning.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3:
Methodology

In this chapter, we examine the basic problem and our approach to its solution. As mentioned in Chapter 1, we consider a scenario where a searcher attempts to locate and maintain observation of a target. We model the target's behavior as a discrete time Markov Chain. The associated state space being the target's location, activity, or any specific attributes that change over time or in some sequential manner. The searcher has only one sensor with which to observe/follow the entity of interest, receiving a reward of one each time step that it detects the current state of the target. In general, the searcher's decision variable is the sensor's next location (i.e., a state of the Markov chain) over time. The objective of the searcher is to allocate the sensor dynamically so as to earn the largest expected total reward over some finite time horizon N, with the current discrete time step being $n \in \mathcal{N} = \{1, 2, \ldots, N\}$. There are four basic settings for this problem, which are listed below. We delve briefly into the first two since they are the simplest settings but focus the majority of our effort on the last two as they are the most insightful.

- Known Transition Dynamics with an Oracle
- Known Transition Dynamics without an Oracle
- Unknown Transition Dynamics with an Oracle (Naïve and Single-Miss methods)
- Unknown Transition Dynamics without an Oracle (hardest and most interesting)

For the last two policies (those without an oracle) we assume that our Markov Chain is irreducible (i.e., the target will never enter a "sink" state or "sink" subset of states). This ensures that it is possible for us to regain observation of the target once we lose track of it. Hence, if we (the searcher) keep looking, at say, state 1, then at some point we will eventually regain observation of the target. We define our state space as $\mathcal{L} = \{1, 2, \ldots, L\}$. Further, we use the following notation to indicate states within the state space: $x, \ell, i, j \in \mathcal{L}$. Of note, Appendix A summarizes and lists the notation used within Chapters 3 and 4.

3.1 Known Transition Dynamics with an Oracle (KTO)

In this first and very basic setting, Known Transition Dynamics with Oracle (KTO), we assume that after each time step, the searcher is given, by the Oracle, the true movement/location of the target. For example, if, at $n = 5$, the searcher looks in state i but the target actually transitioned to state j, the searcher is given that information before moving to the next decision/time step, $n = 6$. This enables us to easily resolve the rewards and choose where to send the sensor for the next time step. Further, we also assume that the underlying transition probabilities are known, making this setting primarily an *exploitation* problem removing the standard *exploration-exploitation* tension. In this setting the searcher always places the sensor on the mode or state with the highest transition probability.

3.2 Known Transition Dynamics with No Oracle (KTNO)

In this section, since the searcher knows the true transition probabilities, we just condition on the last known location of the target and the subsequent sequence of misses to determine the most likely state to which to send or allocate the sensor. In essence, we power up the sub-matrices of P.

More precisely, suppose the target was last seen in location x_1 in period 1, the sensor misses the target in states $x_2, x_3, \ldots, x_{n-1}$. The question for the searcher is: Where to place the sensor in period n given the last known location and the sequence of misses (i.e., the sample path $X_1 = x_1, X_2 \neq x_2, \ldots, X_{n-1} \neq x_{n-1}$)? For $x \in \mathcal{L}$ we consider

$$Pr(X_n = x | X_{n-1} \neq x_{n-1}, \ldots, X_2 \neq x_2, X_1 = x_1) = \frac{Pr(X_n = x, X_{n-1} \neq x_{n-1}, \ldots, X_2 \neq x_2, X_1 = x_1)}{Pr(X_{n-1} \neq x_{n-1}, \ldots, X_2 \neq x_2, X_1 = x_1)}$$

The goal is to find the most likely state in period n, so it suffices to find the maximizer of the numerator above. That is, we want to maximize

$$Pr(X_n = x, X_{n-1} \neq x_{n-1}, \ldots, X_2 \neq x_2, X_1 = x_1) = P_{x_1, x_2^-} \cdot P_{x_2^-, x_3^-} \cdot \ldots \cdot P_{x_{n-1}^-, x}$$

for all $x \in \mathcal{L}$, where P_{x_1, x_2^-} is row x_1 without the element P_{x_1, x_2} of the transition matrix, $P_{x_2^-, x_3^-}$ is the sub-matrix formed by removing row x_2 and column x_3, and $P_{x_{n-1}^-, x}$ is the x'th column of P without the x_{n-1} entry of the transition matrix.

As an example, consider a transition probability matrix over the states $\{1, 2, 3\}$.

$$P = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 0.1 & 0.2 & 0.7 \\ 0.5 & 0.2 & 0.3 \\ 0.8 & 0.1 & 0.1 \end{bmatrix} \end{array}$$

Suppose that $x_1 = 1$. Since the mode of the first row is the third entry, the searcher places the sensor in state 3 in period 2. If the target is not found there, then $X_2 \neq 3$. Using the formula above leads to

$$Pr(X_3 = 1, X_2 \neq 3, X_1 = 1) = [0.1, 0.2] \times \begin{bmatrix} 0.1 \\ 0.5 \end{bmatrix} = 0.11$$

$$Pr(X_3 = 2, X_2 \neq 3, X_1 = 1) = [0.1, 0.2] \times \begin{bmatrix} 0.2 \\ 0.2 \end{bmatrix} = 0.06$$

and

$$Pr(X_3 = 3, X_2 \neq 3, X_1 = 1) = [0.1, 0.2] \times \begin{bmatrix} 0.7 \\ 0.3 \end{bmatrix} = \mathbf{0.13}$$

so the searcher is better off putting the sensor in state 3. As before, if the target is found then we set $X_3 = 3$, otherwise the searcher selects the state corresponding to the largest of

$$Pr(X_4 = 1, X_3 \neq 3, X_2 \neq 3, X_1 = 1) = [0.1, 0.2] \times \begin{bmatrix} 0.1 & 0.2 \\ 0.5 & 0.2 \end{bmatrix} \times \begin{bmatrix} 0.1 \\ 0.5 \end{bmatrix} = 0.041$$

$$Pr(X_4 = 2, X_3 \neq 3, X_2 \neq 3, X_1 = 1) = [0.1, 0.2] \times \begin{bmatrix} 0.1 & 0.2 \\ 0.5 & 0.2 \end{bmatrix} \times \begin{bmatrix} 0.2 \\ 0.2 \end{bmatrix} = 0.034$$

and

$$Pr(X_4 = 3, X_3 \neq 3, X_2 \neq 3, X_1 = 1) = [0.1, 0.2] \times \begin{bmatrix} 0.1 & 0.2 \\ 0.5 & 0.2 \end{bmatrix} \times \begin{bmatrix} 0.7 \\ 0.3 \end{bmatrix} = \mathbf{0.095}$$

Thus, as before, the searcher is better off placing the sensor in state 3 for period 4. The search proceeds along these lines for as long as the target is not found. As it turns out, it is optimal for the searcher to forever place the sensor in state 3 as long as the target is not found, since

$$Pr(X_n = 3, X_{n-1} \neq 3, \cdots, X_3 \neq 3, X_2 \neq 3, X_1 = 1) = [0.1, 0.2] \times \begin{bmatrix} 0.1 & 0.2 \\ 0.5 & 0.2 \end{bmatrix}^{n-3} \times \begin{bmatrix} 0.7 \\ 0.3 \end{bmatrix}$$

is larger than the corresponding computation for states 1 and 2, for any $n \geq 3$. Since the matrix $[0.1, 0.2; 0.5, 0.2]$ is sub-stochastic, it can be seen that $Pr(X_n \neq 3, X_{n-1} \neq 3, \cdots, X_3 \neq 3, X_2 \neq 3, X_1 = 1) \rightarrow 0$, so the target must eventually be found by placing the sensor in state 3. Indeed, leaving the sensor static in (any) single state guarantees that the target is found, as long as the Markov chain is irreducible. Once found in state 3, the searcher is then better off placing the sensor in state 1, and so on.

These ideas can be extended to the case where the searcher has w sensors, $1 \leq w < L$; when $w = L$ the target is always found. The idea is to put the sensors in the most likely states conditioned on the initial known state and the sequence of misses. Specifically, for a sample path $X_1 = x_1, X_2 \neq \mathbf{x}_2, \ldots, X_{n-1} \neq \mathbf{x}_{n-1}$, with $\mathbf{x}_2, \ldots, \mathbf{x}_{n-1} \in \mathcal{L}^w$ the most likely state in period n is found by finding the w maximizers of

$$Pr(X_n = x, X_{n-1} \neq \mathbf{x}_{n-1}, \ldots, X_2 \neq \mathbf{x}_2, X_1 = x_1) = P_{x_1, \mathbf{x}_2^-} \cdot P_{\mathbf{x}_2^-, \mathbf{x}_3^-} \cdot \ldots \cdot P_{\mathbf{x}_{n-1}^-, \mathbf{x}}$$

for all $x \in \mathcal{L}$, where P_{x_1, x_2^-} is row x_1 without the elements $P_{x_1, \mathbf{x}_2}$ of the transition matrix, $P_{\mathbf{x}_2^-, \mathbf{x}_3^-}$ is the sub-matrix formed by removing rows $\mathbf{x}_2$ and columns $\mathbf{x}_3$, and $P_{\mathbf{x}_{n-1}^-, \mathbf{x}}$ is the x'th column of P without the $\mathbf{x}_{n-1}$ entries of the transition matrix.

3.3 Unknown Transition Dynamics with Oracle (UTO)

In this third basic setting, Unknown Transition Dynamics with Oracle (UTO), we assume that after each time step, the searcher is given the true movement/location of the target, again from an Oracle, but this time he has to learn or estimate the true underlying transition probabilities. For example, if, in $n = 5$, the searcher looks in state i but the target actually transitioned to state j, the searcher is given that information before moving to the next decision/time step, $n = 6$. This enables a quick resolution of the rewards and easy calculation of the estimated transition probabilities for the next time step. We explore this setting in two ways. The first is more intuitive and a better method for this setting but is fragile or difficult to extend to our final setting. Therefore, we develop the second approach with a few different variants. As a final note, the developments in this section apply when the searcher has multiple sensors.

Case 1: Full dependence on Constant Oracle

We examine a Markov chain with unknown transition probability matrix, i.e., unknown transition dynamics but with an Oracle that provides the target's location at the beginning of each time step (therefore we are only attempting to predict a single step transition). In this case, we do not need to utilize a MAB approach. Instead due to the Oracle, we can update the elements of the empirical transition matrix at the end of each time step. Hence, the optimal action for the searcher is to place the sensor in the state that is the empirical mode out of the last (known state).

Case 2: Naively Using the Oracle for Reset

While the above setting is intuitive and works quite well, what if the searcher is unable to receive the Oracle's information every time step and instead gets a periodic update (say every 6 or 12 time steps)? In the operational setting this could correspond to the target (terrorist) going home every night or a small fishing boat returning to harbour in the evening. In both of these cases, if a time step is defined as a single hour, the Oracle would provide its periodic update every roughly 12 hours. This motivates the following versions of the UTO setting.

In this approach we estimate the transition probabilities using only the hits' information. Specifically, for each state-pair $(i, j) \in \mathcal{L}^2$ we compute the ratio of the number of hits in state j when the target was just seen in state i (the state $i \to j$ transition) up to time n, $h_{i,j}^{(n)}$, over the total number of sensor placements in (or views of) state j when the target was last in state i up to time n, $v_{i,j}^{(n)}$. Specifically, our initial estimator of $p_{i,j} = P(X_{t+1} = j | X_t = i)$ is

$$\hat{q}_{i,j}^{(n)} = \frac{h_{i,j}^{(n)}}{v_{i,j}^{(n)}}, \tag{3.1}$$

when $v_{i,j}^{(n)} > 0$. Of note, $q_{i,j}^{(n)}$ is the i'th, j'th element of $Q^{(n)}$, the estimated transition probability matrix at time n. Since this estimator does not use miss information, the $\sum_j \hat{q}_{i,j}^{(n)}$ may be strictly smaller than 1. Rescaling each row to sum to 1 produces the following estimator:

$$q_{i,j}^{(n)} = \frac{h_{i,j}^{(n)}}{v_{i,j}^{(n)}} \left(\sum_{\ell \in \mathcal{L}} \frac{h_{i,\ell}^{(n)}}{v_{i,\ell}^{(n)}} \right)^{-1}. \tag{3.2}$$

As an example of these ideas imagine that, prior to rescaling, we have the below row of probability element estimates for a four state Markov chain:

$$\frac{h_{1,\mathcal{L}}^{(n)}}{v_{1,\mathcal{L}}^{(n)}} = \begin{bmatrix} \frac{7}{28} & \frac{2}{16} & \frac{5}{27} & \frac{12}{41} \end{bmatrix}$$

and recall that in this case the oracle discloses the last position of the target. This means that we can execute the below updates at each time step (hence, using the oracle only for reset). Let's say we choose to allocate the sensor to state 4. If the target transitions from its current state, 1, to state 2 we would miss it and update the above row of fractions as follows:

$$\frac{h_{1,\mathcal{L}}^{(n+1)}}{v_{1,\mathcal{L}}^{(n+1)}} = \begin{bmatrix} \frac{7}{28} & \frac{2}{16} & \frac{5}{27} & \frac{12}{\mathbf{42}} \end{bmatrix}$$

If, on the other hand, the target transitions from its current state, 1, to state 4 we would receive a hit and update the above row of fractions as follows:

$$\frac{h_{1,\mathcal{L}}^{(n+1)}}{v_{1,\mathcal{L}}^{(n+1)}} = \begin{bmatrix} \frac{7}{28} & \frac{2}{16} & \frac{5}{27} & \frac{\mathbf{13}}{\mathbf{42}} \end{bmatrix}$$

From this last row of fractions we would update the 1st row of our transition probability estimates, with the rescaled probabilities as follows:

$$q_{i,\mathcal{L}}^{(n)} = \frac{\left[\frac{7}{28} \quad \frac{2}{16} \quad \frac{5}{27} \quad \frac{13}{42}\right]}{\left(\frac{7}{28} + \frac{2}{16} + \frac{5}{27} + \frac{13}{42}\right).}$$

If the searcher can cover the entire state space in each iteration with sensors, then there are no misses, and the estimator defined above is the classical MLE obtained as the solution of

$$\max_{q_{i,1},\ldots q_{i,L}} \prod_{j=1,\ldots,L} q_{i,j}^{h_{i,j}^{(n)}}$$

subject to

$$\sum_j h_{i,j}^{(n)} = \#\text{ transitions out of state } i \text{ by time } n$$

and

$$\sum_j q_{i,j} = 1, \text{ with } q_{i,j} \geq 0.$$

for each $i \in \mathcal{L}$, when the number of transitions out of state i is positive and also where $q_{i,j}^{h_{i,j}^{(n)}}$ is the probability $q_{i,j}$ raised to the $h_{i,j}^{(n)}$'th power. Of course, when $L = 2$ (i.e., there are just two states), a miss gives as much information as a hit; the same is true when the searcher can cover $L - 1$ states with sensors. However, it is not clear how this generalizes to larger state spaces, even when just two states are not searched in a period (e.g., one sensor and $L = 3$).

Case 3: Using the Oracle only for Reset

While maximizing the likelihood of the hits' observations is good, we aren't distributing the density of the target's movement to the numerator of our transition counters when the sensor returns a miss or zero. Remember the example from Case 2 that only updated the denominator for the miss. Intuitively, we know that a miss means the target transitioned to one of the other (unobserved) states but we do not update any of their associated fractions. This means that we are not using all of the information we could glean from each time step. To rectify this situation we maximize the likelihood of both hit and miss observations.

For ease of reading we drop the time step notation on all variables since this calculation is executed during a single time step. Since we are only looking at the single-step misses, the maximum likelihood optimization problem is separable into L optimization problems (below), one for each row of the transition matrix. The resultant likelihood function, capturing all of the hits and single-step misses so far, that we maximize is (abusing notation):

$$L(q_{i,1},\ \ldots\ ,\ q_{i,L}) = \prod_{j=1}^{L} q_{i,j}^{h_{i,j}} \left(1 - q_{i,j}\right)^{(v_{i,j} - h_{i,j})}.$$

For each $i \in \mathcal{L}$ with $v_{i,j} > 0$, we

$$\max_{q_{i,1},\ldots,q_{i,L}} L(q_{i,1},\ \ldots\ ,\ q_{i,L}),$$

with the following constraints:

$$\sum_{j=1}^{L} q_{i,j} \leq 1$$

$$q_{i,j} \geq 0\,, \quad \forall j \in \mathcal{L}$$

Setting $q_{i,j}^{*} = 0$ is optimal when $h_{i,j} = 0$, so henceforth we assume that $0 < h_{i,j} \leq v_{i,j}$. The resultant log-likelihood or objective function is

$$\max_{q_{i,j}} \quad \log L(q_{i,1},\ \ldots\ ,\ q_{i,L}) \ =\ \sum_{j=1}^{L} h_{i,j} \log(q_{i,j}) \ +\ (v_{i,j} - h_{i,j}) \log(1 - q_{i,j}).$$

The objective function is a sum of concave functions, and therefore is concave. The constraint set, being a simplex, is convex. Hence, the Karush-Kuhn-Tucker (KKT) conditions, as listed by Dimitri Bertsekas in [17] and developed by William Karush in [18] and Harold Kuhn and Albert Tucker in [19], are sufficient for optimality. These are

$$\frac{h_{i,j}}{q_{i,j}} \ -\ \frac{v_{i,j} - h_{i,j}}{1 - q_{i,j}} \ =\ k + \lambda_j\,, \quad \forall j \in \mathcal{L}$$

where k is the Lagrange multiplier for the sum constraint, and $\lambda_j \geq 0$ is the Lagrange multiplier for the non-negativity constraint, $q_{i,j} \geq 0$. We have $\lambda_j = 0$ if $q_{i,j} > 0$, when $h_{i,j} > 0$ (true by assumption), so we can disregard these multipliers in the analysis below.

Since the partial derivatives are monotone decreasing with

$$\frac{h_{i,j}}{q_{i,j}} - \frac{v_{i,j} - h_{i,j}}{1 - q_{i,j}} \rightarrow \infty \text{ as } q_{i,j} \rightarrow 0$$

and

$$\frac{h_{i,j}}{q_{i,j}} - \frac{v_{i,j} - h_{i,j}}{1 - q_{i,j}} \rightarrow -\infty \text{ as } q_{i,j} \rightarrow 1.$$

Therefore, all $q_{i,j} > 0$ at optimality since the Markov chain is assumed to be irreducible. Also

$$\frac{h_{i,j}}{q_{i,j}} - \frac{v_{i,j} - h_{i,j}}{1 - q_{i,j}} = 0 \iff q_{i,j} = \frac{h_{i,j}}{v_{i,j}}.$$

so that $q_{i,j} \leq \frac{h_{i,j}}{v_{i,j}}$ if and only if $k \geq 0$. Summing over all the $q_{i,j}$'s for this row we get

$$k \geq 0 \iff \sum_{j=1}^{L} \frac{h_{i,j}}{v_{i,j}} \geq 1.$$

Based on the above results, there are only three possibilities, dependent on the value of the sum of our transition counters for this row, $\sum_{j=1}^{L} \frac{h_{i,j}}{v_{i,j}}$. These three possibilities or cases are enumerated as follows:

1. $\sum_{j=1}^{L} \frac{h_{i,j}}{v_{i,j}} = 1$, in which case the optimal solution is $q_{i,j} = \frac{h_{i,j}}{v_{i,j}}$.
2. $\sum_{j=1}^{L} \frac{h_{i,j}}{v_{i,j}} > 1$, so the optimal solution satisfies the root equation below for $k > 0$.

   Solving for $q_{i,j}$ we get,

   $$\frac{h_{i,j}}{q_{i,j}} - \frac{v_{i,j} - h_{i,j}}{1 - q_{i,j}} = k \iff h_{i,j} - v_{i,j} q_{i,j} = k q_{i,j} - k q_{i,j}^2 \iff k q_{i,j}^2 - (v_{i,j} + k) q_{i,j} + h_{i,j} = 0$$

   for all $j \in \mathcal{L}$.

Hence,

$$q_{i,j} = \frac{v_{i,j} + k \pm \sqrt{(v_{i,j} + k)^2 - 4kh_{i,j}}}{2k}$$

with $k > 0$. The radical is positive, since $v_{i,j} \geq h_{i,j} \geq 0$ leads to

$$(v_{i,j} + k)^2 - 4kh_{i,j} \geq h_{i,j}^2 + 2h_{i,j}k + k^2 - 4kh_{i,j} = (h_{i,j} - k)^2 \geq 0$$

Taking the positive root first, $q_{i,j}^{(+)}$, we get:

$$q_{i,j}^{(+)} = \frac{v_{i,j} + k + \sqrt{(v_{i,j} + k)^2 - 4kh_{i,j}}}{2k} \geq \frac{1}{2}$$

which is not necessarily true. Therefore, we only deal with the negative root, $q_{i,j}^{(-)}$. This boundary condition, $\sum_{j=1}^{L} q_{i,j}^{(-)} = 1$, leads to the root equation in k given by

$$(L-2)k + \sum_{j=1}^{L}(v_{i,j} - \sqrt{(v_{i,j} + k)^2 - 4kh_{i,j}}) = 0,$$

with unique solution k^*, by construction. In conclusion, the MLE's for the transition probabilities are,

$$q_{i,j} = \frac{v_{i,j} + k^* - \sqrt{(v_{i,j} + k^*)^2 - 4k^*h_{i,j}}}{2k^*}.$$

3. In case $\sum_{j=1}^{L} \frac{h_{i,j}}{v_{i,j}} < 1$, with $k^* < 0$ in the root equation, the MLE is the same,

$$q_{i,j} = \frac{v_{i,j} + k^* - \sqrt{(v_{i,j} + k^*)^2 - 4k^*h_{i,j}}}{2k^*},$$

but with $k^* < 0$. The intuition is that the empirical transition probabilities $\frac{h_{i,j}}{v_{i,j}}$, obtained by only counting the hits, are inflated to maximize the likelihood of hits and misses.

In conclusion, this MLE approach, while making use of all the observations, has two major limitations. Computationally, it requires solving a root problem after each new observation and operationally, it presumes the searcher receives a target location update each time period from the oracle. The former can be ameliorated by warm starting the root finding algorithm with the last solution and the latter leads to the next scenario.

Case 4: No Oracle Present

At the other extreme of the gamut, the searcher may never receive feedback from an oracle, making a MLE, including both hits and misses, substantially more difficult. The issue in this setting is that rather than having an optimization problem for each row of the transition matrix, due to the separability structure, there is a single optimization problem involving all the transition probabilities. In this subsection we sketch the ideas for a solution approach.

The starting point is a sample path of hits and misses. Namely, let $\tau_1 = 1$, and $\tau_2, \tau_3, \ldots, \tau_{\zeta_n}$ be the (random) times where the target is located, where $1 \leq \zeta_n \leq n$ is the (random) number of times the target is found by time n. Then a sample path between the first two hits is $X_1 = x_1, X_2 \neq x_2, \ldots, X_{\tau_2-1} \neq x_{\tau_2-1}, X_{\tau_2} = x_{\tau_2}$; between hits two and three $X_{\tau_2+1} \neq x_{\tau_2+1}, X_{\tau_2+2} \neq x_{\tau_2+2}, \ldots, X_{\tau_3-1} \neq x_{\tau_3-1}, X_{\tau_3} = x_{\tau_3}$; and so on.

Inspired by Section 3.2, we maximize the likelihood by time ζ_n (for simplicity)

$$L(q_{1,1}, \ldots, q_{1,L}; \ldots; q_{L,1}, \ldots, q_{L,L})$$

$$= Pr(X_{\tau_{\zeta_n}} = x_{\tau_{\zeta_n}}, X_{\tau_{\zeta_n}-1} \neq x_{\tau_{\zeta_n}-1}, \ldots, X_{\tau_2+1} \neq x_{\tau_2+1}, X_{\tau_2} = x_{\tau_2}, X_{\tau_2-1} \neq x_{\tau_2-1}$$

$$, \ldots, X_2 \neq x_2, X_1 = x_1)$$

$$= Q_{x_1,x_2^-} \cdot Q_{x_2^-,x_3^-} \cdot \ldots \cdot Q_{x_{n-1}^-,x_{\tau_2}} \cdot Q_{x_{\tau_2},x_{\tau_2+1}^-} \cdot \ldots \cdot Q_{x_{\tau_{\zeta_n-1}}^-,x_{\tau_{\zeta_n}}}$$

for all states $x \in \mathcal{L}$, where Q_{x_1,x_2^-} is row x_1 without the element Q_{x_1,x_2} of the estimated transition matrix Q, $Q_{x_2^-,x_3^-}$ is the sub-matrix formed by removing row x_2 and column x_3, and $Q_{x_{\tau_{\zeta_n-1}}^-,x_{\tau_{\zeta_n}}}$ is the $x_{\tau_{\zeta_n}}$'th column of Q without the $x_{\tau_{\zeta_n-1}}^-$ entry. The optimization problem leading to the ML estimator is

$$\max_{q_{i,j},(i,j)\in\mathcal{L}^2} \log L(q_{1,1}, \ldots, q_{1,L}; \ldots; q_{L,1}, \ldots, q_{L,L}),$$

for each state pair (i, j) with $v_{i,j} > 0$, with the following constraints:

$$\sum_{j=1}^{L} q_{i,j} \leq 1, \quad \forall i \in \mathcal{L}$$

$$q_{i,j} \geq 0\,, \quad \forall j \in \mathcal{L}$$

The constraint set, being the intersection of L simplexes, is convex. The Hessian of the objective function can be shown to be negative definite, so that the objective function is concave. Hence, as in Case 3 above, the KKT conditions are sufficient for optimality. Unfortunately, the resulting root equations can't be solved analytically in this case. We believe that an online algorithm, such as online gradient descent, would be useful in this setting, but leave as future work the problem of developing a more insightful approach.

3.4 Exploration and Exploitation

We answer two primary questions in this section. First, why not use the current estimate of the highest transition probability, i.e., the mode of a given row of $Q^{(n)}$ to determine the sensor allocation? Second, how can the searcher incorporate uncertainty in a way that induces efficient exploration of all the states? We tackle these questions in the context of Cases 2 and 3 of the preceding section, for a searcher that has only one sensor.

The main purpose of the resulting search rule is to ensure that the algorithm finds or learns the optimal arm or true mode of a given row of the transition matrix P, instead of narrowing in on a sub-optimal arm. A quick example will help demonstrate the need or motivation for this index policy. Consider the following true transition probability vector for state 1,

$$p_{1,\mathcal{L}} = \begin{bmatrix} \frac{1}{10} & \frac{2}{10} & \frac{3}{10} & \frac{4}{10} \end{bmatrix}.$$

Now, imagine that after a number of periods, say n, the estimate $p_{1,\mathcal{L}}$ is

$$\begin{bmatrix} \frac{1}{5} & \frac{1}{4} & \frac{6}{20} & \frac{1}{4} \end{bmatrix}$$

where $\frac{6}{20}$ should be interpreted as 6 hits in state 3 out of 20 views in state 3 when the target was just seen (or reported by the oracle) in state 1.

If the searcher always selects the mode of this row to place the sensor then, since according to our current estimate it is our best option, we will continue to sample from that state, increasing the precision of this specific transition probability estimate. As n gets larger, the estimate remains close 0.3 with high probability, since that is in fact its true value and we will therefore never explore the rest of the arms of this row since they appear to be worse. But, we know that this is a sub-optimal arm since in fact $p_{1,4} = 0.4$. This small example highlights the need for some form of exploration term to force the algorithm to continue exploring arms that appear to be sub-optimal at the time but that may in fact be optimal, as in the case above.

The first step to derive an index that forces exploration is to bound the estimator error probability, $Pr(|q_{i,j}^{(n)} - p_{i,j}| > \epsilon)$, for $\epsilon > 0$. We now argue how this can be done, for the estimator $q_{i,j}^{(n)}$ in Cases 2 and 3 of Section 3.3. To keep the notation simple we omit the super-script (n) .

For $q_{i,j}$ as in Case 3 of Section 3.3, and $k^* > 0$

$$q_{i,j} - p_{i,j} > \epsilon \iff \frac{v_{i,j} + k^* - \sqrt{(v_{i,j} + k^*)^2 - 4k^* h_{i,j}}}{2k^*} > p_i + \epsilon$$

$$\iff v_{i,j} + k^* - \sqrt{(v_{i,j} + k^*)^2 - 4k^* h_{i,j}} > 2k^*(p_{i,j} + \epsilon)$$

$$\iff (v_{i,j} + k^* - 2k^*(p_{i,j} + \epsilon))^2 > (v_{i,j} + k^*)^2 - 4k^* h_{i,j}$$

$$\iff -4k^*(p_{i,j} + \epsilon)(v_{i,j} + k^*) + (2k^*(p_{i,j} + \epsilon))^2 > -4k^* h_{i,j}$$

$$\iff h_{i,j} - (p_{i,j} + \epsilon)(v_{i,j} + k^*) + k^*(p_{i,j} + \epsilon)^2 > 0$$

$$\iff k^* < \frac{h_{i,j} - v_{i,j}(p_{i,j} + \epsilon)}{(p_{i,j} + \epsilon) - (p_{i,j} + \epsilon)^2}.$$

Hence,

$$Pr(q_{i,j} - p_{i,j} > \epsilon) = Pr\left(k^* < \frac{h_{i,j} - v_{i,j}(p_{i,j} + \epsilon)}{(p_{i,j} + \epsilon) - (p_{i,j} + \epsilon)^2}\right) \tag{3.3}$$

$$\leq Pr\left(h_{i,j} - v_{i,j}(p_{i,j} + \epsilon) > 0\right) \leq \exp(-2v_{i,j}\epsilon^2), \tag{3.4}$$

the last inequality by Hoeffding's Lemma.

The same proof technique applies for $k^* < 0$, leading to

$$Pr(q_{i,j} - p_{i,j} < -\epsilon) \leq \exp(-2v_{i,j}\epsilon^2).$$

Regarding the estimator of Case 2 without rescaling (c.f., Equation 3.1) in Section 3.3 and using the same approach we get

$$Pr(q_{i,j} - p_{i,j} > \epsilon) = Pr\left(\frac{h_{i,j}}{v_{i,j}} - p_{i,j} > \epsilon\right) = Pr\left(h_{i,j} - v_{i,j}(p_{i,j} + \epsilon) > 0\right) \leq \exp(-2v_{i,j}\epsilon^2),$$

and similarly in the case of $k^* < 0$, so we end up with the same bound as Case 3.

From Equations 3.3 and 3.4 we see that the MLE approach is less conservative than the estimator obtained by only considering hits, which is further supported by the numerical work in Chapter 4. A more refined proof technique is needed to flesh out the gain derived by including the miss observations, but this is left as future work.

From here the classical MAB index follows,

$$q_{i,j} + \sqrt{\frac{1.5 \log \sum_{\ell=1}^{L} v_{i,\ell}}{v_{i,j}}}, \tag{3.5}$$

for each $i \in \mathcal{L}$. A constant larger than 1.5 leads to more exploration than needed, while the opposite is true for a constant smaller than 1.5; see [20] for a derivation. The intuition behind Equation 3.5 is that each sub-optimal state is sampled logarithmically with the number of views from the source state, while the best state (i.e., the mode) is sampled the rest of the time.

Given that the target was just observed in state i, the MAB algorithm proceeds by placing the sensor in the state with the largest index,

$$\underset{j=1,\ldots,L}{\operatorname{argmax}} \quad q_{i,j} + \sqrt{\frac{1.5 \log \sum_{\ell=1}^{L} v_{i,\ell}}{v_{i,j}}}.$$

It is well-known (see Theorem 1 in [20]) that this approach produces an expected regret with an upper bound that is logarithmic with the number of views.

To complete the picture, for the estimator of Case 2 with rescaling (c.f., Equation 3.2) in Section 3.3 we get

$$Pr(q_{i,j}-p_{i,j} > \epsilon) = Pr\left(\frac{h_{i,j}}{v_{i,j}}\left(\sum_{\ell\in\mathcal{L}}\frac{h_{i,\ell}}{v_{i,\ell}}\right)^{-1} - p_{i,j} > \epsilon\right) = Pr\left(\frac{h_{i,j}}{v_{i,j}} > \frac{\sum_{\ell\neq j}\frac{h_{i,\ell}}{v_{i,\ell}}}{1-(p_{i,j}+\epsilon)}(p_{i,j}+\epsilon)\right),$$

after working inside the parentheses. Using standard arguments, the right-hand side above becomes

$$\begin{aligned}
&\leq Pr\left(\frac{h_{i,j}}{v_{i,j}} > p_{i,j}+\epsilon\right) + Pr\left(\sum_{\ell\neq j}\frac{h_{i,\ell}}{v_{i,\ell}} < 1-p_{i,j}-\epsilon\right) \\
&\leq Pr\left(\frac{h_{i,j}}{v_{i,j}} > p_{i,j}+\epsilon\right) + \sum_{\ell\neq j} Pr\left(\frac{h_{i,\ell}}{v_{i,\ell}} < p_{i,\ell}-\frac{\epsilon}{p_{i,\ell}}\right) \\
&\leq \exp(-2v_{i,j}\epsilon^2) + \sum_{\ell\neq j}\exp\left(-2v_{i,\ell}\left(\frac{\epsilon}{p_{i,\ell}}\right)^2\right).
\end{aligned}$$

This bound can be used to produce an index similar to Equation 3.5, but with higher regret due to the larger upper bound on the probability of error. However, the numerical results in the next chapter suggest that the rescaled MLE has smaller regret (i.e., better performance) than the unscaled MLE, suggesting that the inequalities above are too loose.

As a final note, while in this section we only considered a searcher with a single sensor, the developments can be extended to a multiple-sensor setting following the ideas of Chen, Wang, and Yuan in their "Combinatorial Multi-Armed Bandit: General Framework, Results and Applications" [21], who study the problem of how to optimally pull several arms simultaneously.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 4:
Analysis and Simulation Results

In this chapter, we examine the initial simulation results, validating the analytic bounds calculated in Chapter 3. Specifically, in a couple small examples we see eight to ten times faster convergence rates when compared to the naïve approach. Of note, we will examine this comparison between the Naïve method and our Single-Miss Algorithm throughout this chapter. Here is a quick outline or summary:

1. Tiny System (4 States)
2. Small System (10 States)

To simulate our algorithm's performance against the Naïve method, we leverage MATLAB to simulate the target's behaviour, based on each specific setting's transistion probabilities, and then implement both methods to attempt to learn the target's behaviour pattern or transition probabilities. Specifically, we capture, at each time step n, the estimated or empirical probability of the target transitioning from state 1 to state 4, or in notation, $q_{1,4}^{(n)}$.

Further, we break the Naïve method into two separate versions shown in the plots as Naïve with normalization and Naïve without normalization. The first, as its name implies, normalizes each row of the transition probability matrix at each time step while the second does not (this breaks the law of total probability but since the algorithm only needs the row of **index** values to determine the next sensor allocation the algorithm still functions). The non-normalized version instead, plots the raw ratio of hits to attempted observations or in notation it defines the estimated probability as: $q_{i,j}^{(n)} = \frac{h_{i,j}^{(n)}}{v_{i,j}^{(n)}}$. The reason for this becomes apparent in Section 4.1. Namely, the normalized version consistently over-estimates the target's probabilites since the normalization process we use treats each estimated ratio as equally precise.

In both Sections 4.1 and 4.2, MATLAB was used to simulate the bahaviour of the target using the standard built-in random number generator with each method being replicated 100 times per method. The mean of these 100 simulations was then used to generate the 95% Confidence Interval bounds denoted by the thin colored lines in the following figures.

4.1 Tiny System (Four States)

In this section we examine simulated convergence rates of both the basic Naïve and our proposed Single-Miss MLE methods on a very small and dense example. Specifically, we examine a four state system corresponding to a potential terrorist's behaviour within a city. The four states being his house, a café, his workplace, and a store. His transition kernel or transition probability matrix is defined as:

$$P = \begin{array}{c} \\ house\ (1) \\ work\ (2) \\ store\ (3) \\ café\ (4) \end{array} \begin{array}{c} \begin{array}{cccc} house\ (1) & work\ (2) & store\ (3) & café\ (4) \end{array} \\ \begin{bmatrix} 0.10 & 0.20 & 0.30 & \mathbf{0.40} \\ 0.90 & 0.02 & 0.04 & 0.04 \\ 0.70 & 0.03 & 0.15 & 0.12 \\ 0.82 & 0.06 & 0.02 & 0.10 \end{bmatrix} \end{array}$$

Since we are looking to learn the behaviour pattern of our target, we measure our performance on the mode of a row. In this case, we compare how quickly the methods can estimate the house to café transition probability, in notation the $p_{1,4} = 0.4$ probability. Of note, the horizontal axis in Figures 4.1 through 4.4 is discrete time. In Figures 4.1 and 4.3, the vertical axis is probability with the thick black line marking the true probability, 0.4.

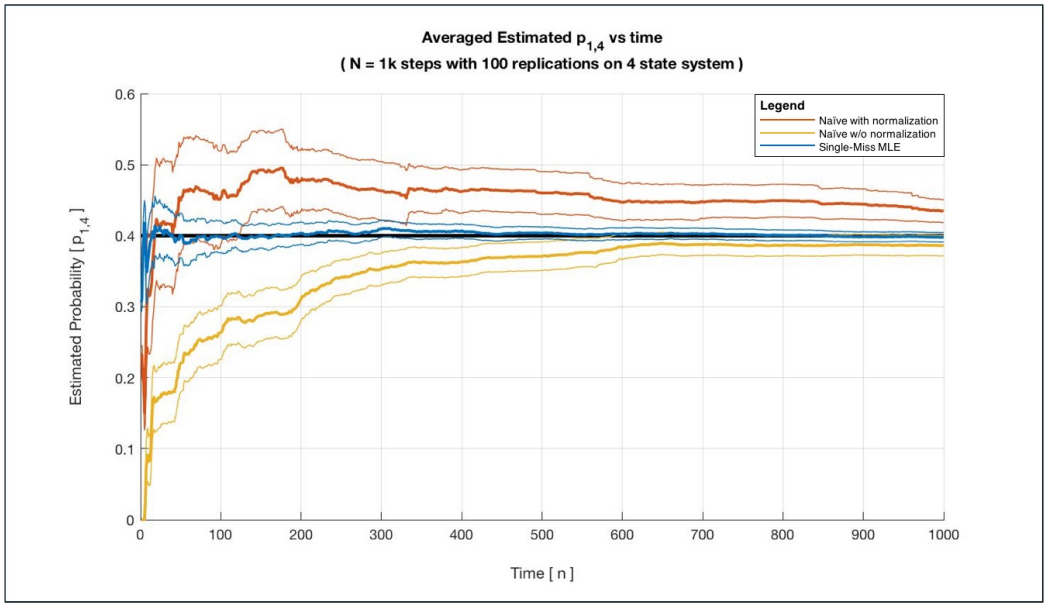

Figure 4.1. Four State Estimated Transition Probabilities vs. Time (mean with 95% confidence intervals). Generated in MATLAB.

As can be seen in Figure 4.1, our Single-Miss MLE method's 95% Confidence Interval captures the true probability well before time step $n = 50$, the Naïve without normalization method captures it around time step $n = 500$, and the Naïve with normalization method doesn't capture it until well after time step $n = 1000$. A quick calculation shows that our Single-Miss MLE method learns at least ten times faster than the better of the naïve versions. Further, we can also see that the normalized Naïve version does in fact over-estimate our parameter of interest. This interesting finding is further examined in Section 4.2.

Additionally, we can see from Figure 4.2 that we also attain logarithmic learning rates with both MLE methods. At the same time though, the Single-Miss MLE method clearly outperforms both Naïve versions in both the expected cumulative regret as well as the 95% Confidence Bounds on that mean. For this analysis, we define the expected regret (vertical axis in Figures 4.2 and 4.4) as the absolute difference between the true probability and the current estimate $q_{i,j}^{(n)}$, with these differences being summed at every time step. In notation:

$$\mathcal{R}_{i,j}^{(N)} = \sum_{n=1}^{N} \left| q_{i,j}^{(n)} - p_{i,j} \right| \text{ which for state 1 to 4 is: } R_{1,4}^{(N)} = \sum_{n=1}^{N} \left| q_{1,4}^{(n)} - 0.4 \right| \tag{4.1}$$

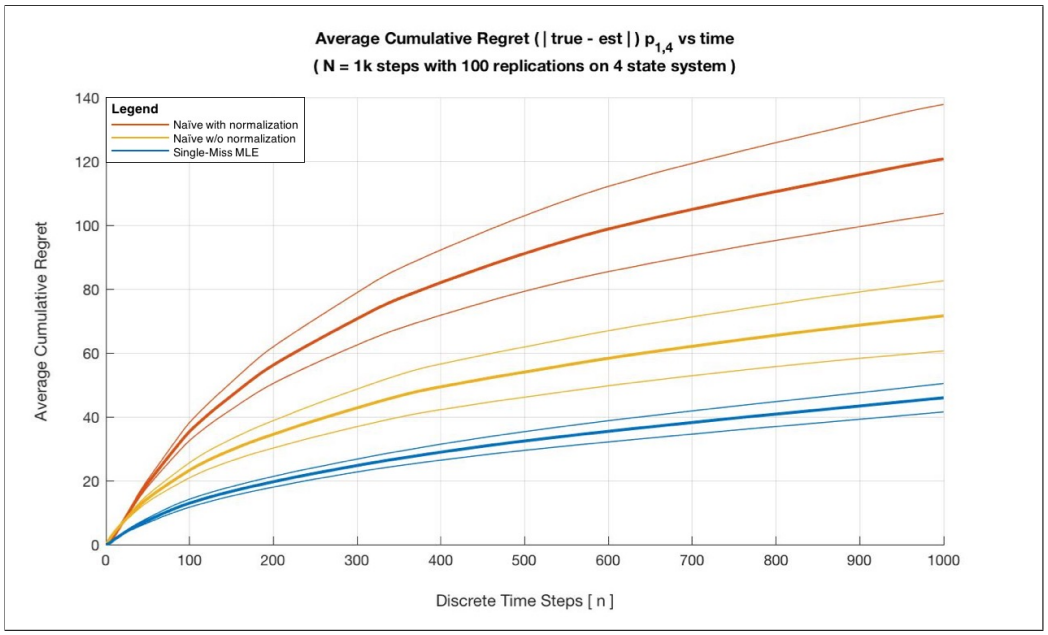

Figure 4.2. Four State Expected Regret vs. Time (mean with 95% confidence intervals). Generated in MATLAB.

4.2 Small System (Ten States)

In this section, we expand the state space from four to ten states. This might correspond to adding another location, maybe a bar, and splitting each of these five states into a day and night version.

Below is a randomly generated transition kernel or transition probability matrix for the target's behaviour modeled with ten states. The result of applying the algorithms to this behaviour pattern are noted in Figures 4.3 and 4.4. Of note in the below behaviour matrix, the $p_{1,4}$ probability is intentionally set to 0.4 for ease of comparison to the four state system analyzed in Section 4.1.

$$P = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \end{array} \begin{array}{c} \begin{array}{cccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{array} \\ \left[\begin{array}{cccccccccc} 0 & 0.16 & 0 & \mathbf{0.40} & 0.13 & 0.10 & 0 & 0.10 & 0 & 0.10 \\ 0 & 0.23 & 0.16 & 0 & 0 & 0.18 & 0 & 0.20 & 0 & 0.22 \\ 0 & 0 & 0.23 & 0.22 & 0 & 0 & 0.28 & 0.26 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.53 & 0 & 0 & 0.46 \\ 0.30 & 0.11 & 0 & 0.11 & 0.13 & 0.12 & 0 & 0 & 0.13 & 0.09 \\ 0.40 & 0 & 0 & 0 & 0 & 0.16 & 0.16 & 0 & 0.10 & 0.17 \\ 0.32 & 0 & 0.11 & 0.12 & 0.16 & 0 & 0.11 & 0 & 0.17 & 0 \\ 0.64 & 0 & 0 & 0.12 & 0 & 0.11 & 0 & 0 & 0.12 & 0 \\ 0.45 & 0.13 & 0 & 0 & 0.11 & 0 & 0 & 0.10 & 0.09 & 0.11 \\ 0.61 & 0 & 0.12 & 0 & 0 & 0.13 & 0 & 0 & 0.13 & 0 \end{array}\right] \end{array}$$

Since we are looking to learn the behaviour pattern of our target, we again measure our performance on the mode of the first row. In this case, we compare how quickly the methods can estimate the state 1 to state 4 transition probability, or in notation $p_{1,4} = 0.4$. The figures in this section following the same format as the previous section, namely the horizontal axis being discrete time. The vertical axis in Figure 4.3 is probability and the vertical axis in Figure 4.4 is the cumulative regret as defined in Equation 4.1.

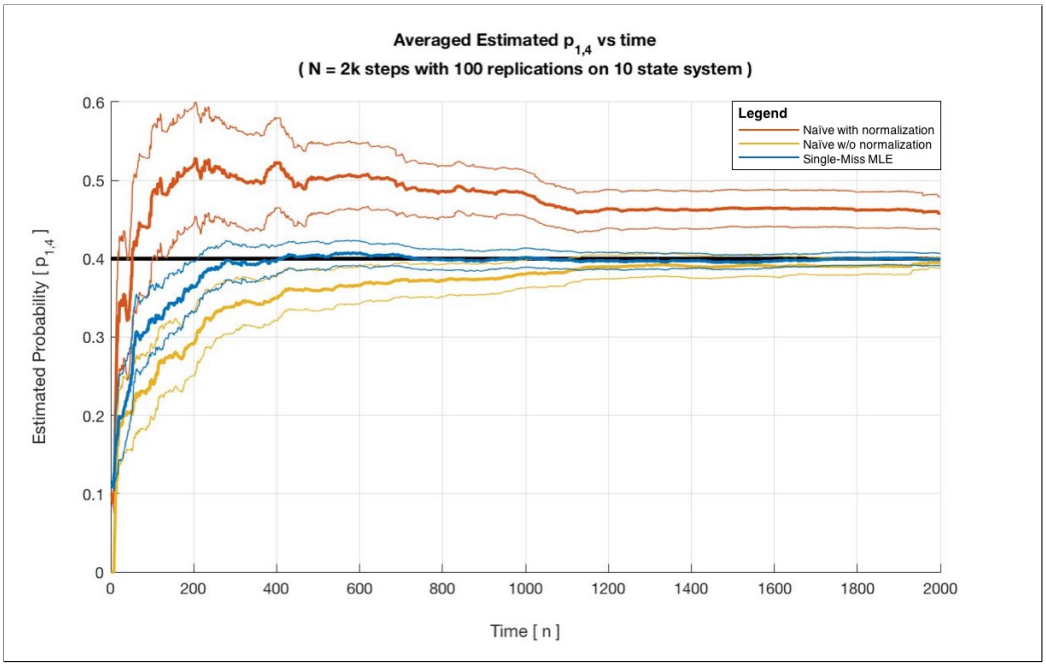

Figure 4.3. Ten State Estimated Transition Probabilities vs. Time (mean with 95% confidence intervals). Generated in MATLAB.

Again, we see the estimation power of the Single-Miss method continues to outperform both Naïve versions in Figure 4.3. In this setting though, the effect of learning from miss data is less pronounced. Knowing that the target did not transition to a given state, a miss, provides less information than a miss does in the four state system, since the probability density is distributed across nine states in this case vice just the three from Section 4.1.

Figure 4.3, highlights even more strongly the fact that the normalized Naïve version remains over-inflated for a very long period of time. This is due to the normalization process failing to account for the different precision of each ratio. Since the "sub-optimal" arms are looked at far less frequently (think far smaller sample size), their hits versus attempted views ratios are by nature less precise (have higher variance) than the more "optimal" arms. The intuition here is that as a sample size increases we decrease the variance of our estimated parameter, a specific transition probability in this case. Therefore, if we normalize without some form of a weighting scheme tied to the different arms' precisions we end up over-inflating the mode as can be clearly seen in Figure 4.3.

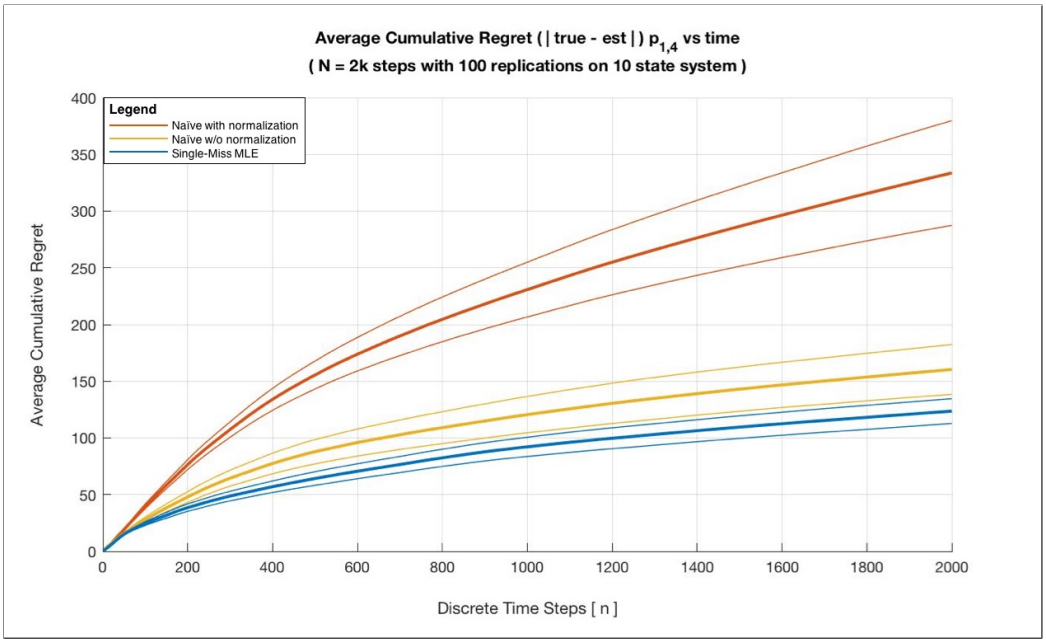

Figure 4.4. Ten State Expected Regret vs. Time (mean with 95% confidence intervals). Generated in MATLAB.

Figure 4.4 shows that the logarithmic learning rates or cumulative expected regrets are still achieved as the state space increases. We also see the very pronounced effect of over-inflation on the cumulative regret for the normalized Naïve method. And lastly, we again see that the Single-Miss MLE method continues to outperform both Naïve versions.

CHAPTER 5:
Conclusion and Recommendations

This chapter provides a quick summary of or conclusion to this research as well as exploring some areas of recommended future research or work.

5.1 Conclusions

This thesis presents a novel method for leveraging and applying Machine Learning methods and techniques to learning behaviour patterns modeled as discrete time Markov Chains. Further, the Single-Miss MLE algorithm obtains a logarithmic learning rate and performs significantly better in expectation than the Naïve methods described in Chapters 3 and 4 in the scenarios examined and presented in Chapter 4. At the same time though, given an appropriate exploration inflation term, examined and developed in Chapter 3, the Single-Miss MLE and Naïve MLE methods all obtain the desired logarithmic rate.

Chapter 1 laid out the background to this thesis and examined some different areas of research that intersect in an interesting way. Specifically, leveraging current Stochastic Multi-Armed Bandit theory as well as modeling a target's behaviour as a discrete time Markov Chain. The intersection of these two fields of research in our problem and our resulting Single-Miss MLE algorithm is novel as far as the writers are aware.

5.2 Future Work

This section explores some proposed future work or extensions. These ideas being naturally tied to or flowing from assumptions and limitations mentioned in Chapters 1 and 3.

Leveraging Multi-Step Misses

The convergence rate gains obtained by leveraging the complimentary miss data, namely the Single-Miss approach versus the Naïve approach, resulted in a significant performance increase. Therefore, extracting additional data from the multi-step miss sequences or strings should likewise yield additional performance improvements.

Impact of Density and Increased State Space Size

As Chapter 4 highlights, the increased state space size resulted in a decreased effect or benefit from gleaning information from miss data. This indicates that further analysis should be conducted exploring the impact or effect of the state space's size and resultant transition probability matrix density on the learning rate.

Leveraging a Bayesian Prior

Depending on how one defines the state space, some transitions may be completely impossible while others are just highly unlikely. An example of the former is a potential terrorist transitioning from a café at 8 am to the airport at midnight in a single one hour time step which is impossible. Others may be highly unlikely based on current technology or some prior analysis of the target's behaviour by the searcher. These therefore, provide the motivation for enabling or leveraging a Bayesian approach to this problem which would enable this prior knowledge to be captured.

Noisy Sensor Responses

What if the sensor returns a noisy response, think false negatives or positives? This might correspond to a patrol officer walking by the outside of a café during the morning rush. The patrol officer glances through the window but is unable to see everyone in the café. The officer then must give some sort of estimate of their certainty of the target's absence from the café. The converse could also occur where a sensor thinks it observed the target but in fact did not. Implementing noisy responses from the sensor would decrease the abstraction of this problem and hence, increase the applicability of the resultant algorithm.

Multiple Sensors

Given a single target, what if the Law Enforcement Agency, the searcher, had multiple sensors to deploy or allocate? This concept could provide an interesting extension to the current Single-Miss MLE method where the number of sensors is still less than the total number of observable states.

Multiple Related Targets

What if instead of multiple sensors, there were multiple targets that were somehow either spatially correlated or dependent? This would enable the algorithm to leverage data from multiple targets, potentially increasing the learning rate on both targets.

Time-Horizon Weighting

One key assumption of this thesis is that the target's behaviour pattern remains stationary. In other words, the target's behaviour does not change with time. What if we relax this assumption and allow the target's behaviour to vary with time? This motivates the development of a time-horizon weighting scheme or method which values more recent observations over older data. This could also then be extended to form the basis for a change point detection algorithm.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A:
Basic Mathematical Notation

While not absolutely necessary, this appendix provides a quick-reference guide to the notation we use in Chapters 3 and 4.

Sets

$\ell, i, j \in \mathcal{L} = \{1, 2, \ldots, L\}$. Let $L = |\mathcal{L}|$ (total number of different states). And, let ℓ^- be the state space complement of ℓ or all other states in $\mathcal{L}$.

$n \in \mathcal{N} = \{1, 2, \ldots\}$, being the current discrete time step. We also use k as a past time step.

Data

T^n, the target's current location or state at time step n (n, the current time step).

P, true, underlying, transition probability matrix governing the target's transition behavior.

$p_{i,j}$, target's probability of transitioning from state i to j, corresponding to the (i, j)'th element within the matrix P.

Variables

S^n, sensor's current location or where it was sent at time n (n, the current time step).

$v_{i,j}^{(n)}$, cumulative, by n, number of sensor allocations to j, given the target came from i.

$h_{i,j}^{(n)}$, cumulative, by n, number of target observations in j, given the target came from i.

$q_{i,j}^{(n)}$, target's estimated, as of n, probability of transitioning from state i to j corresponding to the (i, j)'th element within the matrix $Q^{(n)}$, defined next.

$Q^{(n)}$, current, by n, empirical transition probability matrix estimating the true behavior.

$\mathcal{R}_{i,j}^{(n)}$, cumulative regret, by n, between the estimated and true transition probabilities for the transition i, j. Specifically, we define this regret as: $\mathcal{R}_{i,j}^{(n)} = \sum_{k=1}^{n} |q_{i,j}^{(k)} - p_{i,j}|$

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B:
Simple Markov Chain Example

While this appendix also is not completely necessary, it is included to serve as a very quick introduction to discrete time Markov chains for those readers who do not have exposure to these concepts and ideas. Towards this goal, we take a moment to define a Markov Chain for the non-technical reader as well as provide a simple example to facilitate understanding.

Layman's Definition: A Markov Chain is a set of states or **state space** with associated transition probabilities that models an object's stochastic or uncertain behaviour over time. It assumes that future transitions are only dependent on the current state of the object and therefore independent of the past. We call this independence the **memoryless property** of a Markov Chain or the **Markov Property**. While this may seem like a rather large assumption to make, if needed, we can embed information about the past into the current state thereby maintaining this assumption without losing the mathematical power of the memoryless property of Markov Chains. The following is a more succint definition using mathematical notation:

Mathematical Definition: A Discrete Time Markov Chain is a stochastic (think uncertain) process $X_t : t = 1, 2, \ldots$ taking values in a discrete **state space** $S = \{1, 2, \ldots, s\}$, that satisfies the **Markov Property**. Meaning, for $A \subseteq S$:

$$P(X_{t+1} \in A \mid X_1, \ldots, X_t) \;=\; P(X_{t+1} \in A \mid X_t)$$

Example: A Washing Machine

The simplest example is a washing machine. It is either working or broken, up or down, with these states defining its state space. The set $\{up, down\}$ being this state space. We model the washing machine as a stochastic process $\{X_t, t = 0, 1, \ldots\}$ with X_t taking on the value of either up or $down$ in each time period t. Generically, if $X_t = x$, then the process or machine is said to be in state x at time t. Further, we define the finite state space as $S = \{1, \ldots, s\}$ which corresponds to $\{up, down\}$ in our example. Next, we define the fixed probability of the machine transitioning from its current state in this time period to

another state (or back to the same state) via $p_{i,j}$. In mathematical and probability notation: $P(X_{t+1} = j \mid X_t = i) = p_{i,j}$. Therefore, if the machine is currently up, then we indicate the probability of it going down in the next time period as $p_{up,down}$. We also denote the probability of the machine staying up as $p_{up,up}$. Further, by the Law of Total Probability, $p_{up,up} + p_{up,down} = 1$ or in layman's terms, the machine must be either up or down in the next time period. This last can be expressed in the following two properties:

1. For all i, j; $\quad 0 \leq p_{i,j} \leq 1 \qquad$ (Probabilities must be between zero and one)
2. For all i; $\quad \sum_{j=1}^{m} p_{i,j} = 1 \qquad$ (Law of Total Probability)

Further, we can mathematically express the entire transition dynamics of our washing machine process as a matrix of probabilities. We use the same notation with $p_{i,j}$ corresponding to the i'th, j'th element of our 2-dimensional transition probability matrix, P. In our example it could look like the following:

$$P = \begin{bmatrix} p_{up,up} & p_{up,down} \\ p_{down,up} & p_{down,down} \end{bmatrix} = \begin{bmatrix} .75 & .25 \\ .80 & .20 \end{bmatrix}$$

The above matrix captures the behavior of our simple washing machine example. But as you are no doubt thinking, all breakdowns are not equal. Some might be for minor damage while some might be of a much more extensive nature. This can also be captured by a Markov Chain, as can be seen in the below modified matrix. This method of expanding the state space to capture more detail is very useful and will enable us to apply the algorithms developed within this thesis to a much broader set of problems than would seem possible at first glance. As can be seen, we now have a couple zero probabilities that correspond to the machine breaking down due to minor damage and then somehow becoming broken for major damage (whether this is zero or not though ultimately depends on the system being modeled).

$$P_{mod} = \begin{bmatrix} p_{up,up} & p_{up,down_{min}} & p_{up,down_{maj}} \\ p_{down_{min},up} & p_{down_{min},down_{min}} & p_{down_{min},down_{maj}} \\ p_{down_{maj},up} & p_{down_{maj},down_{min}} & p_{down_{maj},down_{maj}} \end{bmatrix} = \begin{bmatrix} .75 & .20 & .05 \\ .80 & .20 & 0.0 \\ .40 & 0.0 & .60 \end{bmatrix}$$

List of References

[1] K. Lin, "DRAFT: Probability models for practitioners," 2016, unpublished.

[2] E. Hazan, "Introduction to online convex optimization," *Foundations and Trends in Optimization*, vol. 2, no. 3-4, pp. 157–325, 2016.

[3] A. Mahajan and D. Teneketzis, *Multi-armed Bandit Problems*. Boston, MA: Springer Science & Business Media, 2008, ch. 6, pp. 121–151. Available: http://dx.doi.org/10.1007/978-0-387-49819-5_6

[4] S. Agrawal and N. Goyal, "Analysis of Thompson Sampling for the Multi-armed Bandit problem," in *Proceedings of the 25th Annual Conference on Learning Theory*, 2012, vol. 23, pp. 39.1–39.26.

[5] J. L. Devore, *Probability and Statistics for Engineering and the Sciences*. Boston, MA: Cengage Learning, 2015.

[6] E. L. Lehmann and G. Casella, *Theory of Point Estimation*. Boston, MA: Springer Science & Business Media, 1998.

[7] H. Cramér, *Mathematical Methods of Statistics (PMS-9)*. Princeton, NJ: Princeton University Press, 2016, vol. 9.

[8] C. R. Rao, "Information and the accuracy attainable in the estimation of statistical parameters," in *Breakthroughs in statistics*. Springer, 1992, pp. 235–247.

[9] S. Bubeck, N. Cesa-Bianchi *et al.*, "Regret analysis of stochastic and nonstochastic Multi-armed Bandit problems," *Foundations and Trends in Machine Learning*, vol. 5, no. 1, pp. 1–122, 2012.

[10] Y. Costica, "Optimizing classification in intelligence processing," M.S. thesis, Operations Research Department, Naval Postgraduate School, Monterey, CA, 2010. Available: https://calhoun.nps.edu/handle/10945/4986

[11] Y. Nevo, "Information selection in intelligence processing," M.S. thesis, Operations Research Department, Naval Postgraduate School, Monterey, CA, 2011. Available: https://calhoun.nps.edu/handle/10945/10660

[12] D. R. Ellis, "Algorithms for efficient intelligence collection," M.S. thesis, Operations Research Department, Naval Postgraduate School, Monterey, CA, 2013. Available: https://calhoun.nps.edu/handle/10945/37621

[13] M. Tekin, "New perspectives on intelligence collection and processing," M.S. thesis, Operations Research Department, Naval Postgraduate School, Monterey, CA, 2016.

[14] J. Marshall, "Models of intelligence operations," Ph.D. dissertation, Department of Mathematics and Statistics, Lancaster University, 2016.

[15] A. Hepworth, "A Multi-Armed Bandit approach to superquantile selection," M.S. thesis, Operations Research Department, Naval Postgraduate School, Monterey, CA, 2017.

[16] J. Grant, "DRAFT: Optimal partition based search," Ph.D. dissertation, Department of Mathematics and Statistics, Lancaster University, 2017, unpublished.

[17] D. P. Bertsekas, *Nonlinear Programming*. Belmont, MA: Athena Scientific, 1999.

[18] W. Karush, "Minima of functions of several variables with inequalities as side constraints," M.S. thesis, Department of Mathematics, University of Chicago, Chicago, IL, 1939.

[19] H. Kuhn and A. Tucker, "Nonlinear programming," in *Proceedings of 2nd Berkeley Symposium*. Berkeley, CA: Berkeley: University of California Press, 1951, pp. 481–492.

[20] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the Multi-armed Bandit problem," *Machine Learning*, vol. 47, no. 2-3, pp. 235–256, 2002.

[21] W. Chen, Y. Wang, and Y. Yuan, "Combinatorial Multi-Armed Bandit: General framework, results and applications," in *Proceedings of the 30th International Conference on Machine Learning*, 2013, pp. 151–159.

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California